PET/CBM

GUIDA ALL'USO

VOL.1

Adam Osborne
Carrol S. Donahue

EDIZIONE ITALIANA

GRUPPO
EDITORIALE
JACKSON

PET/CBM

GUIDA ALL'USO

VOL.1

Adam Osborne
Carrol S. Donahue

EDIZIONE ITALIANA

GRUPPO
EDITORIALE
JACKSON

© Copyright per l'edizione originale McGraw-Hill Inc. 1980
© Copyright per l'edizione Italiana McGraw-Hill Inc.1984

Il Gruppo Editoriale Jackson ringrazia per il prezioso lavoro svolto nella stesura dell'edizione italiana la signora Francesca Di Fiore, e l'Ing. Roberto Pancaldi.

Traduzione italiana a cura dell'Ing. Enrico Odetti.

Tutti i diritti sono riservati. Stampato in Italia. Nessuna parte di questo libro può essere riprodotta, memorizzata in sistemi di archivio, o trasmessa in qualsiasi forma o mezzo, elettronico, meccanico, fotocopia, registrazione o altri senza la preventiva autorizzazione scritta dell'editore.

Fotocomposizione: Lineacomp S.r.l. - Via Rosellini, 12 - 20124 Milano

Stampato in Italia da:
S.p.A. Alberto Matarelli - Milano - Stabilimento Grafico

PREFAZIONE

Quest'opera, presentata in due volumi, è una guida all'uso dei calcolatori CBM. Oltre ai calcolatori CBM vengono ampiamente descritte anche le unità periferiche di registrazione dei dischetti, i "drive", quelle per le cassette magnetiche e le stampanti.

La presentazione del linguaggio BASIC CBM è molto dettagliata e facile. Il lettore viene "guidato" dalle prime e più semplici nozioni di programmazione dei calcolatori CBM fino a quelle più complesse e sofisticate. Tutte le istruzioni del BASIC CBM sono illustrate dapprima con facili esempi esplicativi poi con programmi più complessi, ma molto ben documentati. Il lettore troverà, oltre alla descrizione dei calcolatori CBM, anche molte utilissime notizie e informazioni che riguardano i personal computer in generale.

Il primo volume di questa guida contiene nei capitoli 1 e 2 l'introduzione ai calcolatori CBM e i loro modi d'impiego. Il capitolo 3 tratta l'elaborazione di testi o "editing". Il capitolo 4 è dedicato ai fondamenti di programmazione in BASIC CBM. Il capitolo 5 sviluppa ulteriormente la programmazione dei calcolatori CBM.

L'impiego delle unità periferiche, la descrizione del Sistema CBM e le appendici sono riportati nel secondo volume.

SOMMARIO

Prefazione III

CAPITOLO 1 - INTRODUZIONE AI CALCOLATORI CBM 1

Modelli CBM 2
Caratteristiche CBM 6
Accensione. 8
Tastiere dei calcolatori CBM 8000 e CBM 2001/B. 16
Tastiera del calcolatore PET 2001/N. 19
Tastiera del calcolatore PET 2001/8K. 22
Unità a disco CBM. Dischetti "floppy". 33
Le stampanti CBM. 42

CAPITOLO 2. IMPIEGO DEI CALCOLATORI CBM 51
Modo immediato 51
La tastiera in modo immediato. Calcoli aritmetici. Movimenti del cursore.
Modo differito 60
Caricamento del programma. Insiemi di caratteri standard e alternativo.
Impiego delle unità a cassette 65
Impiego delle unità a disco 70
Come caricare un programma da dischetto con il BASIC < 3.0. Come caricare un programma da dischetto con il BASIC 4.0.
Impiego delle stampanti CBM 80

CAPITOLO 3. ELABORAZIONE DI TESTI "EDITING" 83

Elaborazione del testo sulla linea corrente. Elaborazione di testi racchiusi tra virgolette. Elaborazioni (editing) delle righe di un programma. Sviluppi dell'elaborazione dei testi nel BASIC 4.0.

CAPITOLO 4. PROGRAMMAZIONE DEI CALCOLATORI CBM 93

Modi di programmazione: immediata e differita. Programmi con una sola linea in modo immediato.
Elementi di un linguaggio di programmazione 98
Numeri di linea. Dati. Operatori. Comandi BASIC.
Istruzioni BASIC 122
Commenti. Istruzioni di assegnazione. Istruzioni di salto. Istruzioni di controllo a ciclo. Subroutine (o sottoprogrammi). Istruzione di ingresso e uscita. Istruzioni PEEK e POKE. Istruzioni END e STOP.
Funzioni 143
Funzioni aritmetiche. Funzioni di stringa. Funzioni del Sistema. Funzioni definite dall'utente.

CAPITOLO 5. CARATTERISTICHE DEI CALCOLATORI CBM 149
Caratteristiche hardware 149
Buffer della tastiera.
Concatenazione di stringhe 153
Stringhe grafiche. Stringhe numeriche.
Programmazione di ingresso e uscita 157
Istruzione PRINT. Movimenti del cursore. Funzione CHR$. Programmazione dei caratteri in ASCII. Ingresso di dati. Programmazione del display e delle stampanti.
Programmazione matematica 194
Addizione. Sottrazione. Moltiplicazione.
Grafica 226
Animazione.
Orologio in tempo reale 235
Numeri a caso 242
Seme dei numeri a caso. Caratteri forzati a caso sullo schermo con POKE.

CAPITOLO 1

INTRODUZIONE AI CALCOLATORI CBM

Questo libro descrive i seguenti calcolatori Commodore:

1. Il PET 2001/8K
2. Il PET 2001/8N, 2001/16N e 2001/32N
3. Il CBM 2001/16B e 2001/32B
4. La serie CBM 4000
5. La serie CBM 8000.

Nel 1977 la Commodore Business Machines presentò il primo calcolatore della serie il PET 2001 (Personal Electronics Transactor). IL PET 2001 è un'unità compatta comprendente una tastiera grafica e una unità a cassette magnetiche. Il CBM 2001, che fu presentato successivamente, ha una tastiera grafica completa. Sebbene funzionalmente eguale al PET, il CBM 2001 e i successivi modelli, non hanno l'unità a cassette magnetiche assemblata al suo interno. Essi dipendono invece da periferiche esterne per memorizzare informazioni. Il calcolatore commerciale CBM 2001/B è una variazione del CBM 2001. La maggior differenza fisica tra il CBM 2001 e il CBM 2001/B concerne la tastiera; il CBM 2001 ha una tastiera con simboli grafici completa, mentre il CBM 2001/B ha una tastiera tipo macchina da scrivere standard senza simboli grafici sui tasti. Il CBM 8016 e il CBM 8032 sono calcolatori commerciali presentati più di recente. Ambedue hanno un display video a 80 colonne e per il resto sono equivalenti al CBM 2001/B. Il CBM 8032 e il CBM 8016 sono identici, salvo che il primo ha una memoria doppia rispetto al secondo.

La Commodore ha presentato anche stampanti e unità a disco. Vengono inoltre continuamente aggiornati il BASIC Commodore e il software del sistema operativo a disco.

Il primo calcolatore della Commodore è stato il PET che divenne subito molto famoso. I calcolatori recentemente introdotti portano invece il nome CBM. **Per brevità in questo libro si adotterà la convenzione di chiamare tutta la linea di calcolatori con il nome CBM salvo che non sia espressamente richiamato il nome originale PET.** Le unità CBM 8016 e CBM 8032 saranno indicate con il numero generico CBM 8000 salvo che non sia diversamente richiesto.

Attualmente i calcolatori CBM sono disponibili con memoria di 8 K, 16 K o 32 Kbyte. Solamente il primo PET aveva una opzione per una memoria di 4 Kbyte. 1 K significa 1024 che equivale a 2^{10}. Un byte contiene un carattere. Le indicazioni di 8 K, 16 K e 32 K si riferiscono alla memoria disponibile per operazioni di lettura e

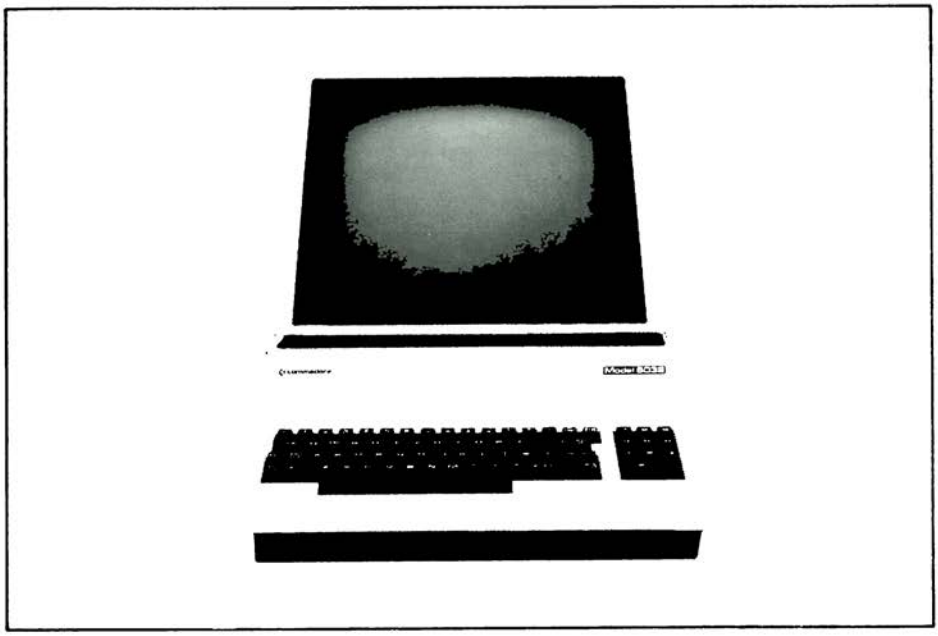


Figura 1-1: Calcolatore CBM 8000

scrittura. Ogni calcolatore CBM ha un'altra memoria che non è accessibile dall'utente. È molto importante conoscere quanta memoria di lettura/scrittura è disponibile. Un calcolatore CBM con più memoria può eseguire programmi più lunghi e gestire anche più dati.

MODELLI CBM

Il CBM 8000 (CBM 8016 e 8032)

Il modello CBM 8000 è indicato in Figura 1-1. La sua caratteristica più notevole riguarda il grande schermo a 80 colonne, detto anche display a Tubo a Raggi Catodici (CRT). Ha una tastiera completa tipo macchina da scrivere, alcuni tasti speciali per l'"editing" sullo schermo e una tastiera solo numerica sulla destra della macchina. Il CBM 8016 ha 16 Kbyte di memoria di lettura/scrittura (non è però disponibile in Italia). Il CBM 8032 ha invece 32 Kbyte di memoria di lettura/scrittura. Si vede quindi che il numero di modello CBM si correla con la dimensione della memoria di lettura/scrittura disponibile. Per completare un sistema commerciale è necessario aggiungere una unità a cassette magnetiche o una unità a disco. È probabile che sia necessaria anche una stampante.

Il CBM 2001/B

Il CBM 2001/B, come il CBM 8000, è un calcolatore commerciale; vedi Figura 1-2. Il display del modello CBM 2001/B è largo 40 colonne e cioè metà della larghezza del display del modello CBM 8000. Il CBM 2001/B ha una tastiera completa con tasti per l'editing sullo schermo e tastiera solo numerica sulla destra. Il CBM 2001/B è disponibile con 16 K o 32 Kbyte di memoria di lettura/scrittura. Come il CBM 8000 anche il CBM 2001/B avrà probabilmente bisogno di una unità esterna a cassette magnetiche o a disco e forse anche di una stampante.

Il PET 2001/N

La serie PET 2001/N è una versione migliorata del calcolatore originale PET, vedi Figura 1-3. Il display è identico a quello del 2001/B. Il PET 2001/N si distingue dal calcolatori commerciali per i tasti con simboli grafici presenti sulla sua tastiera. Il PET 2001 è disponibile con memoria di lettura/scrittura di 8 K (/8N), 16 K (/16N) o 32 K (/32N) byte. Il PET 2001/N e il CBM 2001/B hanno le stesse caratteristiche di collegamento con le periferiche esterne come cassette, dischi o stampanti.

Figura 1-2: Calcolatore CBM 2001/B

Figura 1-3: Calcolatore PET 2001/N

Figura 1-4: Calcolatore PET 2001/8K

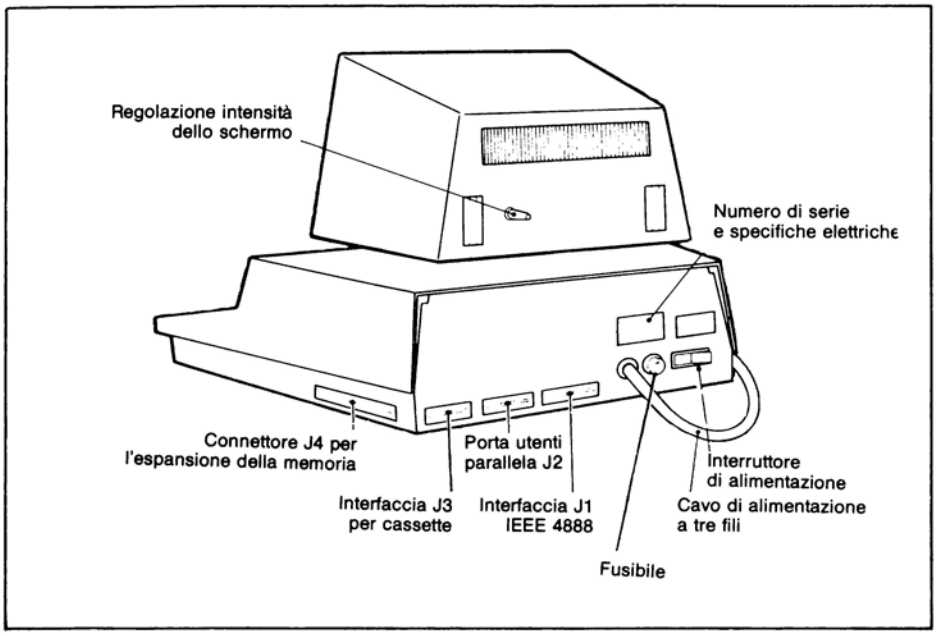


Figura 1-5: Vista del retro di un calcolatore CBM

Il PET 2001/8 K

Il PET 2001/8 K è stato il primo calcolatore presentato dalla Commodore Business Machines. Tutti i modelli CBM successivi sono una evoluzione dell'originale PET 2001/8 K. Il PET 2001/8 K, pur avendo lo stesso display del CBM 2001, si distingue per la sua tastiera compatta e colorata. **I simboli grafici sono rappresentati anche sui tasti** (vedi Figura 1-4).

Grazie alla ridotta dimensione della tastiera, una unità a cassette magnetiche è assemblata sulla sinistra del calcolatore stesso. Il PET 2001/8 K è l'unico modello che comprende una cassetta al suo interno, ma questo comporta lo svantaggio di avere una tastiera più piccola e meno comoda da usare. Il PET 2001/8 K ha 8 Kbyte di memoria di lettura/scrittura. Sono però disponibili espansioni della memoria a 16 K e 32 K. Il PET, come tutti i calcolatori CBM, contiene anche una memoria di sola lettura ("Read Only Memory" ROM) che non è accessibile dall'utente. Questa memoria ROM contiene i programmi permanenti che personalizzano lo specifico modello di calcolatore. **Molti calcolatori PET hanno una "vecchia personalità", dovuta ad una vecchia serie di memorie ROM.**

Una seconda unità a cassette esterna può essere collegata ad un calcolatore PET.

Una stampante o un disco può essere collegato a un PET 2001/8 K solo se contiene memorie ROM con revisione di livello 3.

CARATTERISTICHE CBM

PANNELLO POSTERIORE

Ogni interruttore, connettore o interfaccia è posto sul retro del calcolatore CBM. La Figura 1-5 mostra il retro del calcolatore CBM con la descrizione delle singole parti. È importante conoscere la posizione e la funzione delle singole parti così da non danneggiare il calcolatore per uso errato delle connessioni.

INTERRUTTORE DI ALIMENTAZIONE

L'interruttore di alimentazione è posto sul lato sinistro del pannello posteriore esso è un interruttore tipo "rocker" a due posizioni. Premendolo sul lato esterno si accende il calcolatore, premendolo invece sul lato interno si spegne.

CAVO DI ALIMENTAZIONE

Il cavo di alimentazione a 3 fili connette il calcolatore CBM ad una presa elettrica a corrente alternata.

Si raccomanda di verificare che la tensione applicata sia quella prevista per la corretta alimentazione del calcolatore, (per l'Europa 220 V e 50 Hz).

INTERFACCIA IEEE 488

L'interfaccia IEEE 488 (punto J 1 dello schema) permette al calcolatore CBM di comunicare con le periferiche esterne. Mediante il cavo IEEE si connette appunto una stampante o un disco all'interfaccia IEEE 488.*

PORTE UTENTE PARALLELE (USER PORT)

Questa interfaccia (punto J2 dello schema) può essere usata al posto del connettore IEEE 488 per collegare periferiche al calcolatore CBM. Non è necessario conoscere il suo funzionamento. Se una periferica la dovesse usare, vi sarà fornita l'opportuna documentazione per effettuare il collegamento.

---

* Per una descrizione dettagliata dell'interfaccia IEEE 488 si veda "PET and the IEEE 488 Bus (GPIB)" di E. Fisher e C.W. Jensen edito da Osborne/McGraw-Hill, 1980.

INTERFACCIA PER CASSETTE MAGNETICHE

Questa interfaccia è stata progettata unicamente per unità esterne a cassette (punto J3 dello schema). Essa è posta sulla destra ed è facilmente individuabile perchè di piccole dimensioni.

CONNETTORE PER L'ESPANSIONE DI MEMORIA

È posto sul lato posteriore destro (punto J4) e non è necessario conoscerlo in dettaglio. È possibile espandere la memoria di lettura/scrittura di un calcolatore CBM tramite questo connettore.

MANOPOLA PER LA REGOLAZIONE DELLA LUMINOSITA'

Questa manopola controlla la luminosità dello schermo del display. Provate a ruotare la manopola tutta a sinistra e poi tutta a destra, osservando lo schermo, e così vedrete variare la luminosità e brillantezza dei caratteri.

DISPLAY VIDEO

Il display video è simile allo schermo di un televisore bianco e nero (con persistenza bianca per i vecchi modelli e verde per i nuovi), ma con una ben maggiore risoluzione. Questo significa che si possono vedere piccole immagini e caratteri con maggior chiarezza.

A seconda del modello si possono avere 1000 o 2000 singole posizioni dei caratteri divise in 25 righe di 40 caratteri o 25 righe di 80 caratteri. I caratteri sono a loro volta formati da punti di una matrice di 8 x 8 punti. Vedere la Figura 1-6.

I diversi tipi di display sono descritti separatamente. Se avete un CBM 8000 leggete la seguente descrizione e saltate poi al paragrafo riguardante le tastiere. Se

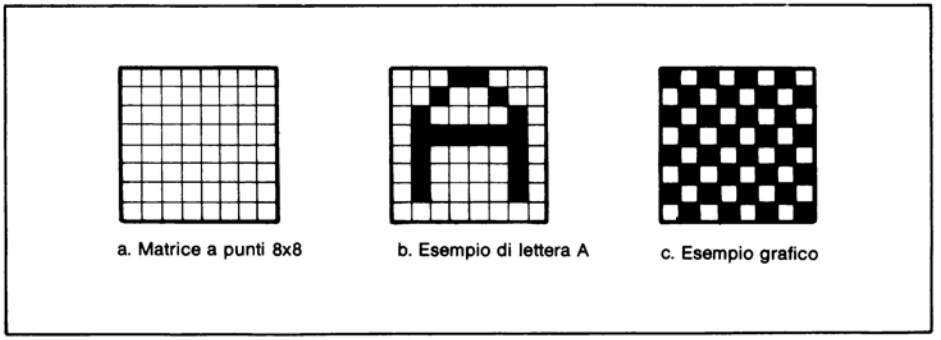


Figura 1-6: Matrice a punti 8x8.

avete invece un CBM 2001/B o un 2001 o un PET saltate invece il seguente paragrafo sul CBM 8000.

CBM 8000

Il display video del CBM 8000 differenzia il modello CBM 8000 dagli altri modelli. Lo schermo è diviso in 2000 spazi eguali ripartiti in 25 righe di 80 caratteri l'una. Ad ogni spazio corrisponde un carattere a cui è associato un byte della memoria.

Sullo schermo si possono visualizzare caratteri alfabetici, numerici, simboli speciali e grafici. Il CBM 8000 normalmente visualizza caratteri alfabetici sia minuscoli che maiuscoli mediante un insieme di caratteri che noi chiameremo "insieme alternativo".

C'è anche in insieme di caratteri standard che permette di rappresentare caratteri grafici, ma non caratteri minuscoli.

CBM 2001/B, PET 2001/N, PET 2001/8 K

I display video dei modelli CBM 2001/B, PET 2001/N e PET 2001/8 K sono praticamente gli stessi. Il display è diviso in 1000 spazi ripartito in 25 righe di 40 caratteri l'una. Ogni carattere è visualizzato in uno spazio.

Tutti i modelli hanno due tipi di caratteri: alfanumerici (alfabetici, numerici e caratteri speciali) e simboli grafici. **Il modello CBM 2001/B, come tutti i modelli CBM 8000, visualizza l'insieme di caratteri alternativo con lettere minuscole e maiuscole; i simboli grafici fanno parte dell'insieme standard.**

Il PET visualizza normalmente l'insieme di caratteri standard che comprende le maiuscole alfanumeriche e i simboli grafici. Lettere minuscole e maiuscole fanno parte dell'insieme di caratteri alternativo.

ACCENSIONE DEL CALCOLATORE

Per accendere il calcolatore CBM procedete come segue:

1. Inserite il cavo di alimentazione in una presa di corrente dotata di messa a terra. *Non tentate di inserire la spina in una presa di corrente non dotata di messa a terra oppure di rimuovere lo spinotto di terra!* Se il calcolatore non è opportunamente collegato a terra può ricevere uno shock elettrico.
2. Accendete il calcolatore premendo sul lato esterno l'interruttore di alimentazione posto sul retro.
3. Attendete l'avviso di READY. Dopo qualche secondo dall'accensione appariranno i seguenti messaggi sullo schermo:

```
### COMMODORE BASIC ###

XXXXX BYTES FREE

READY
▒
```

- Queste quattro linee hanno il seguente significato:

### COMMODORE BASIC ###	Indica che il linguaggio BASIC è stato attivato.
XXXXX BYTES FREE	Indica quanta memoria è disponibile all'utente: 3071 (o un numero simile) nel caso di calcolatore PET con 4 K, 7167 (o un numero simile) nel caso di CBM con 8 K, 15359 (o un numero simile) per un CBM con 16 K, 31743 (o un numero simile) per un CBM con 32 K.
READY	Il calcolatore è pronto per ricevere informazioni.
▒	Il cursore è posizionato sullo schermo dove apparirà il primo carattere che sarà battuto sulla tastiera.

Se non si ottengono i messaggi sopra indicati spegnete il calcolatore e accendetelo nuovamente dopo pochi secondi. Lo schermo potrà apparire pieno di caratteri a caso per un secondo o più. Ciò è normale ogni volta che si spegne e si accende il calcolatore dopo un brevissimo tempo: non preoccupatevi!

GRUPPI DI TASTI DELLE TASTIERE CBM

La tastiera dei calcolatori CBM è impiegata per l'ingresso di istruzioni, di programmi e di dati. Il tipo di tastiera dipende dal tipo di calcolatore CBM. Salvo poche eccezioni, gli stessi tasti sono presenti sia nelle tastiere compatte che in quelle normali. Alcuni tasti si trovano in posizione diversa a seconda del modello di calcolatore.

I tasti delle tastiere CBM possono essere suddivisi come segue: tasti alfabetici, tasti numerici, tasti per simboli speciali, tasti grafici, tasti per funzioni e tasti per il controllo del cursore.

Tasti alfabetici

I tasti alfabetici forniscono le 26 lettere dell'alfabeto inglese, dall'A alla Z. Minuscole e maiuscole sono presenti in tutti i calcolatori CBM.

Tasti numerici

I tasti numerici forniscono le cifre 0,1,2,3,4,5,6,7,8 e 9.

Tasti per simboli speciali

Simboli e caratteri speciali possono comprendere segni di interpunzione e altri simboli comunemente usati. Per esempio il punto, la virgola, il "+" dell'addizione, il "–" della sottrazione, ecc. Altri caratteri universalmente impiegati sono il segno del dollaro "$", il segno di percentuale "%", ecc.

Alcuni caratteri rappresentano una specifica operazione o hanno un preciso significato in una istruzione BASIC. Vedere per una spiegazione dettagliata il capitolo 4.

Tabella 1-1: Tasti per i caratteri grafici.

Linee Orizzontali

#	Superiore
E	3/4 Superiore
D	2/3 Superiore
C	Medio
@	Quasi Medio
F	2/3 Inferiore
R	3/4 Inferiore
$	Inferiore

Linee Verticali

%	Sinistra
T	3/4 Sinistra
G	2/3 Sinistra
B	Quasi Medio
]	Medio
H	2/3 Destra
Y	3/4 Destra
,	Destra

Barra Sottile

7	Superiore
/	Inferiore
4	Sinistra
*	Destra

Barra Spessa

8	Superiore
9	Inferiore
5	Sinistra
6	Destra

Mezzo Blocco

/	Sinistra
//	Inferiore

Triangolo Solido

)	Sinistro Superiore
←	Destro Superiore

Quarto di Blocco Solido

>	<	Sinistro Superiore / Destro Superiore
:	,	Sinistro Inferiore / Destro Inferiore
?		Diagonale

Quarto di Blocco Aperto (Angolo)

=	–	Sinistro Superiore / Destro Superiore
•	Ø	Sinistro Inferiore / Destro Inferiore

Angoli

O	P	Sinistro Superiore / Destro Superiore
L	:	Sinistro Inferiore / Destro Inferiore

Angoli Rotondi

U	I	Sinistro Superiore / Destro Superiore
J	K	Sinistro Inferiore / Destro Inferiore

Semi delle Carte

A	S	Picche / Cuori
Z	X	Quadri / Fiori

T

1	Superiore
2	Inferiore
3	Sinsitro
+	Destro

Simboli

V	X
[	Croce
N	Diagonale Destro
M	Diagonale Sinistro

Griglia

&	Piena
\	Mezza Sinistra
(	Mezza Inferiore

Cerchi

W	Solido
Q	Esterno

Tasti grafici

La tastiera CBM comprende ben 62 simboli grafici ottenuti con la posizione "shift" dei tasti. È possibile quindi creare disegni molto sofisticati.

I caratteri grafici, con la loro rispettiva denominazione, sono elencati nella Tabella 1-1. Simboli simili sono raggruppati assieme così da rendere facilmente comprensibili alcune opzioni grafiche. Si noti che il quadrato, che racchiude i simboli grafici nella Tabella 1-1, non fa parte del simbolo stesso. Il quadrato è stato aggiunto infatti per dare un riferimento spaziale al simbolo.

Tasti per funzioni

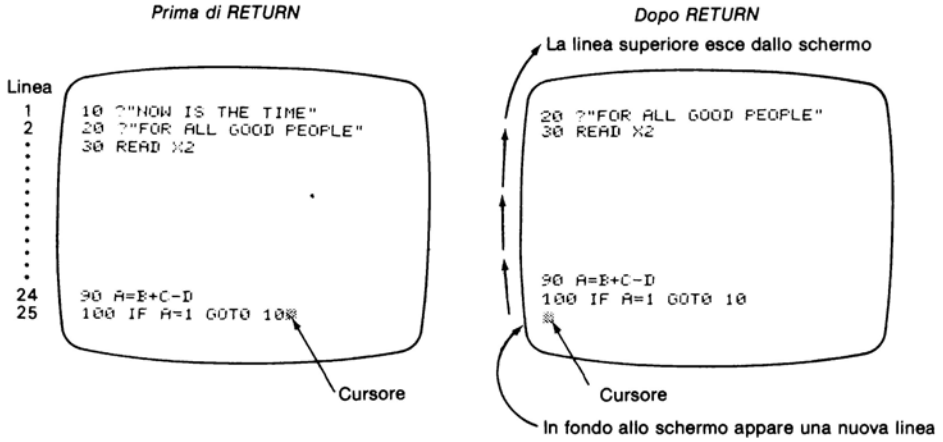
SHIFT. Il tasto di shift si può premere contemporaneamente ad ogni altro tasto e permette di commutare tra i caratteri inferiori e quelli superiori della tastiera. Tutti i tasti forniscono caratteri diversi a seconda che la tastiera sia in posizione "shift" o "non-shift" (caratteri superiori o inferiori) eccetto il tasto ESC. Se SHIFT non è premuto si hanno i caratteri inferiori se premuto invece si hanno i caratteri superiori.

Per maggior praticità sulla tastiera sono presenti due identici tasti SHIFT: uno in basso a sinistra ed uno in basso a destra.

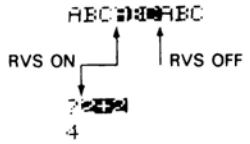
SHIFT LOCK (solo per la tastiera grande). Questo tasto è posizionato sopra il tasto SHIFT di sinistra. Esso va premuto sino allo scatto e permette di bloccare ambedue i tasti SHIFT nella posizione bassa così da lasciar libere le mani per battere altri caratteri come in una normale macchina da scrivere. Per sbloccare il tasto SHIFT LOCK basta premerlo una seconda volta.

RETURN. Ha la funzione equivalente a quella di ritorno del carrello, cioè di andare a capo nelle normali macchine da scrivere. **Quando viene battuto il cursore va a capo nella prima posizione di sinistra dello schermo.**

Quando si è sull'ultima riga in basso, questo comando fa muovere tutto il testo verso l'alto e cioè la prima riga in alto esce dallo schermo mentre in basso si crea una nuova riga vuota alla cui sinistra si porta il cursore.



REVERSE ON/OFF. Questo comando permette di invertire la parte chiara con la parte scura di un carattere sullo schermo. Nello stato REVERSE un carattere appare come il negativo di una fotografia. Normalmente questo tasto è nella posizione spento "off" e per attivarlo basta premerlo con SHIFT non abbassato. Il carattere che successivamente batterete apparirà con il campo invertito e così tutti gli altri di seguito sino a che premerete REVERSE OFF (questa volta con SHIFT abbassato) oppure darete un comando RETURN.



Nota: RETURN fa terminare l'inversione del campo

RUN/STOP. STOP è la funzione non shift di questo tasto. **Il comando STOP ferma l'esecuzione di un programma eventualmente in corso e ristabilisce il collegamento tra la tastiera e il calcolatore.** Provate a vedere come il comando STOP funziona caricando questo piccolo programma, fatto da una sola riga, senza cercare di capire il suo significato. La parte ombreggiata è quella che dovete battere. Appena battete RETURN il programma entra in esecuzione e sul display appariranno in colonna i numeri da 1 verso 100. Premendo STOP vi accorgerete che la numerazione si blocca e il cursore ritorna sullo schermo.

```
FOR I=1 TO 100:?I:NEXT I
 1
 2
 3
 4
 5
 6
 7
 8
 9
 10 ←—— Premere il tasto STOP

BREAK
READY.
```

Se il calcolatore non è in fase di esecuzione il comando STOP non ferma ovviamente niente.

RUN è la seconda parte di questo tasto e funziona con il comando SHIFT abbassato. **Esso carica e pone in esecuzione un programma registrato su una cassetta o un dischetto.**

Tasti di controllo del cursore

Sono quattro tasti con denominazione inglese di: CLEAR SCREEN/HOME, CURSOR UP/DOWN, CURSOR LEFT/RIGHT e INSERT/DELETE.

"Home" è la posizione di "riposo" del cursore ed è in alto a sinistra sullo schermo; il comando HOME deve essere dato con SHIFT non abbassato.

Prima di HOME

```
60 QT=QT+A
70 NEXT X
80 CLOSE1
90 END
1000 POKE 59411,53
1010 T=TI
1020 IF (TI-T)<10 GOTO 1020
1030 POKE 59411,61
1040 QT=0
```

Cursore

Dopo HOME

```
60 QT=QT+A
70 NEXT X
80 CLOSE1
90 END
1000 POKE 59411,53
1010 T=TI
1020 IF (TI-T)<10 GOTO 1020
1030 POKE 59411,61
1040 QT=0
```

Cursore in HOME

CLEAR SCREEN (con SHIFT abbassato) svolge la stessa funzione sul cursore di HOME, ma in aggiunta cancella tutto lo schermo.

Prima di CLEAR SCREEN

```
60 QT=QT+A
70 NEXT X
80 CLOSE1
90 END
1000 POKE 59411,53
1010 T=TI
1020 IF (TI-T)<10 GOTO 1020
1030 POKE 59411,61
1040 QT=0
```

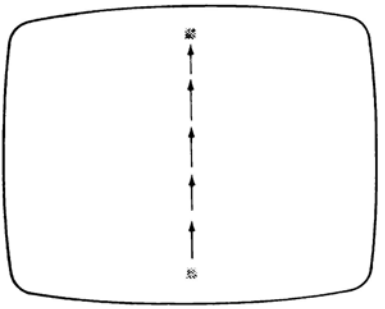
Cursore

Dopo CLEAR SCREEN

Cursore in HOME, lo schermo è pulito

CURSOR UP (con SHIFT abbassato) **muove il cursore di una riga verso l'alto sulla stessa verticale.**

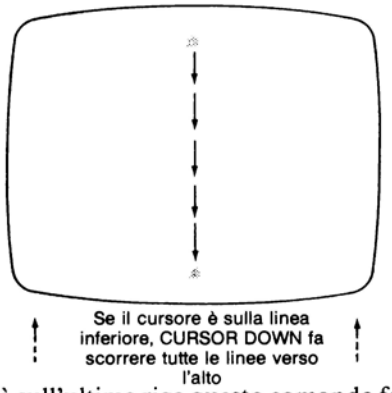
CURSOR UP



Se il cursore si trova già sulla prima riga questo comando non ha effetto. Il cursore muovendosi non cambia o modifica il carattere su cui si sovrappone.

CURSOR DOWN (con SHIFT non abbassato) **muove il cursore sulla stessa colonna verso il basso.**

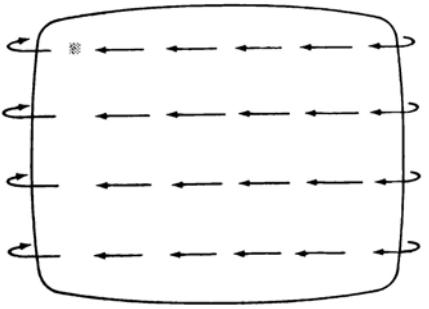
CURSOR DOWN



Se il cursore si trova già sull'ultima riga questo comando fa slittare ("sroll") tutto il testo in alto di una riga.

CURSOR LEFT (SHIFT abbassato) **muove il cursore verso sinistra sulla stessa riga orizzontale.**

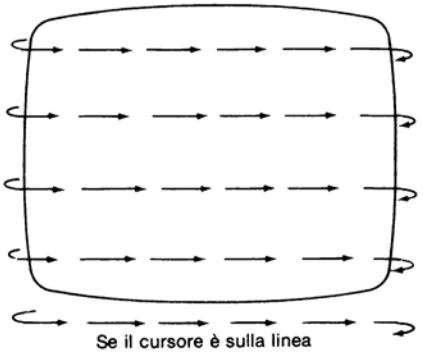
CURSOR LEFT



Questo comando **non ha effetto se il cursore è nella posizione "home".**

CURSOR RIGHT (SHIFT non abbassato) **muove invece il cursore verso destra sulla stessa riga.**

CURSOR RIGHT



Se il cursore è sulla linea inferiore, CURSOR RIGHT fa apparire una nuova linea in basso

Sia CURSOR LEFT che CURSOR RIGHT hanno la proprietà di risalire o discendere rispettivamente da una riga all'altra quando arrivano ad una estremità della riga stessa.

Se il cursore si trova all'ultima posizione in basso a destra il comando CURSOR RIGHT fa scorrere di una riga in alto tutto il testo ("scroll") e porta il cursore all'inizio dell'ultima riga.

I comandi CURSOR LEFT/RIGHT sono usati per scrivere sopra un testo ed equivalgono allo spostamento del carrello in una normale macchina da scrivere.

Il comando **DELETE** si ottiene con il tasto INSERT/DELETE e SHIFT non abbassato. **Esso cancella il carattere immediatamente a sinistra del cursore e muove di un posto verso sinistra tutti i caratteri che sono a destra del cursore compreso.**

```
NOW IS THE TIME▒
NOW IS THE TIM▒
NOW IS THE TI▒
```

Il comando **INSERT** corrisponde al tasto INSERT/DELETE come il precedente, ma con SHIFT abbassato. **Esso inserisce un nuovo spazio libero e in tale posizione potrà essere battuto un nuovo carattere.**

```
NOW IS▒THE TIME
NOW IS▒ THE TIME
NOW IS▒  THE TIME
```

È opportuno precisare che i calcolatori CBM considerano un testo come una sequenza di righe lunghe ciascuna al massimo 80 caratteri. Non è possibile quindi cancellare oltre l'inizio di una riga ne inserire caratteri nuovi così che una riga diventi più lunga di 80 caratteri.

LE TASTIERE DEI CALCOLATORI CBM 8000 E CBM 2001/B

I calcolatori CBM 8000 e CBM 2001/B hanno tastiere del tutto simili a quelle di una normale macchina da scrivere ed hanno inoltre una piccola tastiera solo numerica sulla destra come si può vedere nella Figura 1-7. Tali tastiere possono essere usate in due modi diversi e produrre due distinti insiemi di caratteri.

Chiameremo i due insiemi di caratteri: standard e alternativo.

L'insieme di caratteri alternativo è sempre funzionante appena si accende il calcolatore. Esso produce, con SHIFT non abbassato, le lettere minuscole e con SHIFT abbassato le lettere maiuscole e tutti i caratteri superiori.

Figura 1-7: La tastiera dei calcolatori CBM 8000 e 2001/B

L'insieme standard, che si ottiene invece dando un comando speciale al calcolatore, produce con SHIFT non abbassato le lettere maiuscole e con SHIFT abbassato i caratteri grafici. **Per attivare quest'insieme di caratteri è necessario battere il comando:**

POKE 59468,12 *(RETURN)*

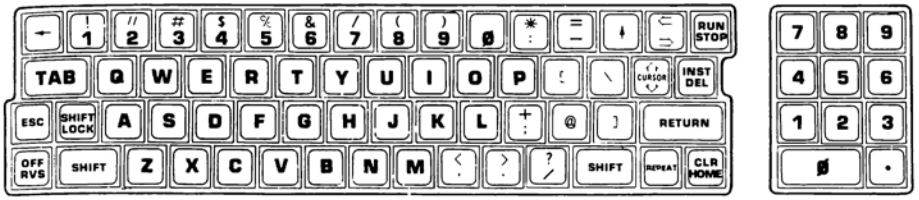
che cambierà immediatamente l'insieme di caratteri. **Per ritornare all'insieme alternativo** date un diverso numero nel comando POKE:

POKE 59468,14 *(RETURN)*

Salvo un diverso avviso riteniamo che sia sempre possibile usare i caratteri standard. Come vedremo meglio in seguito, sappiate inoltre che **ogni istruzione deve terminare con il comando RETURN.**

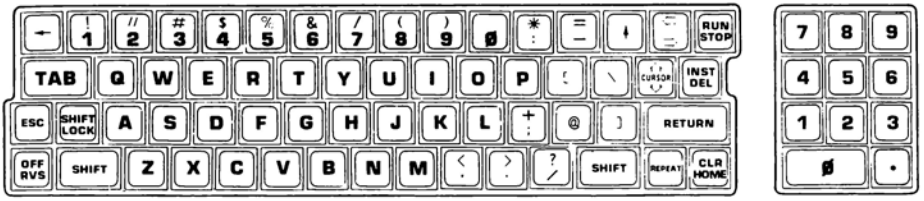
Descriviamo ora le tastiere e poniamo in risalto i vari gruppi di caratteri tratteggiandoli sui disegni.

Tasti alfabetici



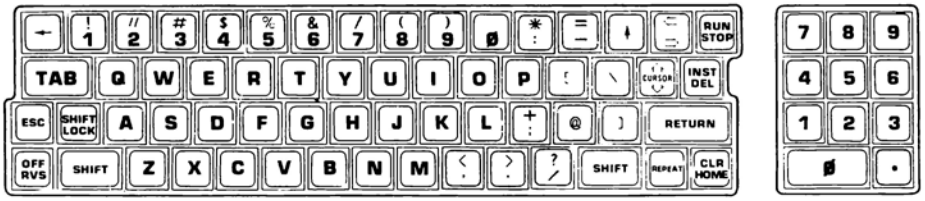
I tasti alfabetici sono quelli ombreggiati nella illustrazione qui sopra. Nel modo alternativo essi producono: lettere minuscole (nel caso non SHIFT) e lettere maiuscole (nel caso SHIFT). Nel modo standard producono invece: lettere maiuscole (nel caso non SHIFT) e simboli grafici (nel caso SHIFT).

Tasti numerici



I tasti numerici sono presenti due volte: sulla tastiera principale (ombreggiati) e sulla piccola tastiera di destra. I primi richiedono di non abbassare il tasto SHIFT mentre i secondi possono essere usati sia nel modo alternativo che standard. Per praticità il tasto del 5, della tastiera piccola, ha una piccola protuberanza.

Tasti per simboli speciali

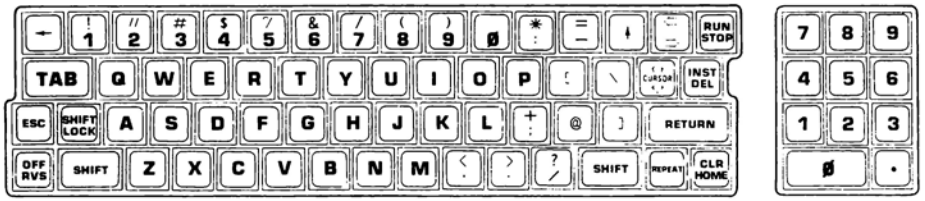


Sono quelli ombreggiati nell'illustrazione qui sopra. I simboli riportati sulla metà superiore dei tasti si ottengono con SHIFT abbassato e con la tastiera in modo alternativo. Gli altri si ottengono come è usuale con SHIFT abbassato o non abbassato.

Tasti grafici

I simboli grafici, pur essendo disponibili, non sono disegnati sulla tastiera dei calcolatori commerciali. **Per ottenerli dovete dare il comando per il modo standard POKE 59468,12 e premere il tasto SHIFT.** Nel disegno di sopra sono ombreggiati quei tasti che corrispondono appunto ai simboli grafici nei calcolatori commerciali.

Tasti per funzioni



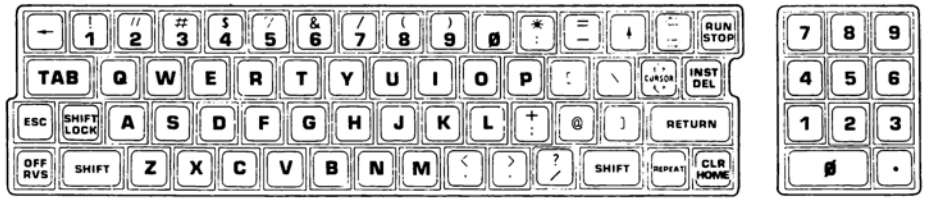
I calcolatori CBM 8000 e 2001/B hanno tre tasti per funzioni oltre a quelli già descritti nel paragrafo "Gruppi di tasti delle tastiere CBM": TAB, ESCAPE (ESC) e REPEAT.

TAB: è il tasto per assegnare e togliere le posizioni di tabulazione ed inoltre per saltare alla colonna di tabulazione successiva.

ESC: è presente solo nella serie CBM 8000. Può essere usato in due casi: per annullare gli effetti dei comandi di inserimento, di inversione dei caratteri o di altre condizioni di ingresso dei testi; oppure è impiegato con altri tasti per produrre funzioni speciali di elaborazione dei testi (come vedremo nel capitolo 5).

REPEAT: permette la ripetizione di ogni altro tasto con cui è premuto contemporaneamente.

Tasti di controllo del cursore



Questi gruppi sono già stati descritti nel paragrafo di pagina 13.

LA TASTIERA DEL CALCOLATORE PET 2001/N

Il calcolatore PET 2001/N ha una tastiera di dimensioni normali, ma con la caratteristica di avere i simboli grafici disegnati sulla superficie anteriore dei tasti (vedi Figura 1-8). Anche questa tastiera presenta i due modi di lavoro: standard e alternativo. **Il modo standard si ottiene subito all'accensione del calcolatore.** Per ottenere il modo alternativo dovete dare il comando:

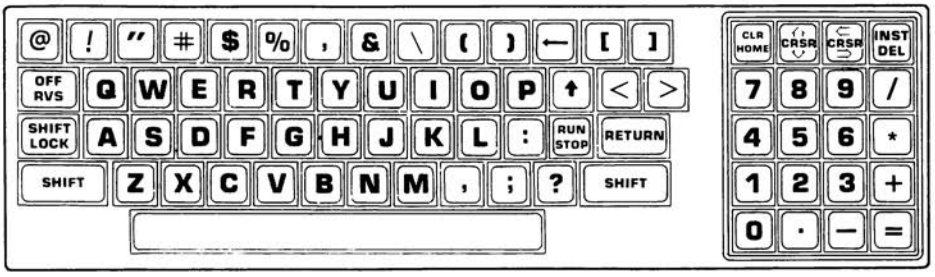
```
POKE 59468,14
```

Per ritornare al modo standard dovete inserire invece:

```
POKE 59468,12
```

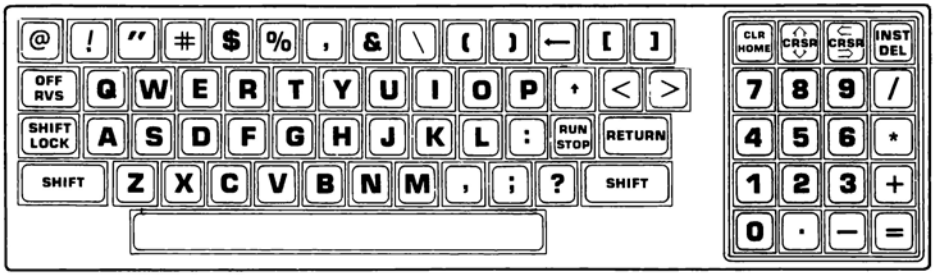
Figura 1-8: La tastiera del calcolatore PET 2001/N

Tasti alfabetici



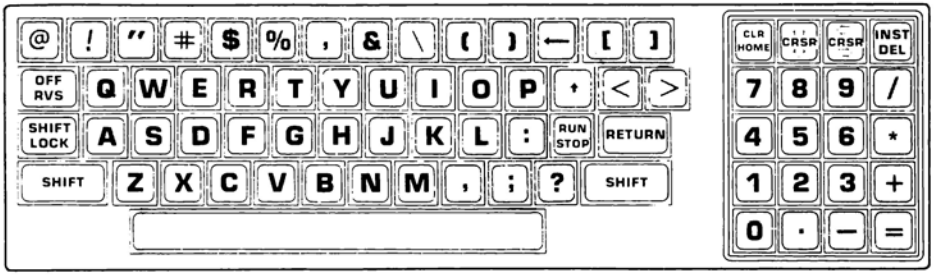
I tasti alfabetici, ombreggiati nel disegno, producono: lettere maiuscole nel modo standard e SHIFT non abbassato; nel modo alternativo producono invece sia le lettere maiuscole che le minuscole a seconda della posizione di SHIFT.

Tasti numerici



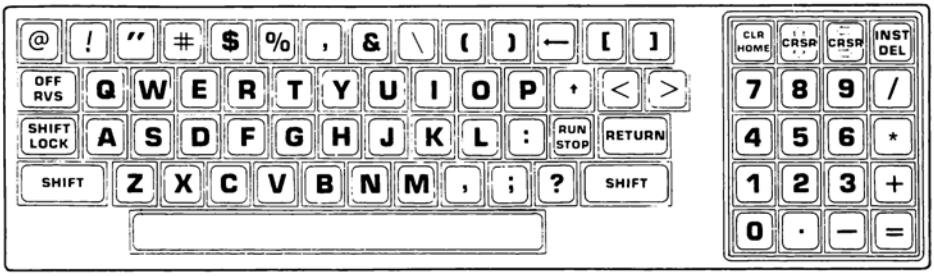
I tasti numerici sono presenti solo sulla tastiera piccola a destra. I numeri sono prodotti solo con il tasto SHIFT non abbassato.

Tasti per simboli speciali



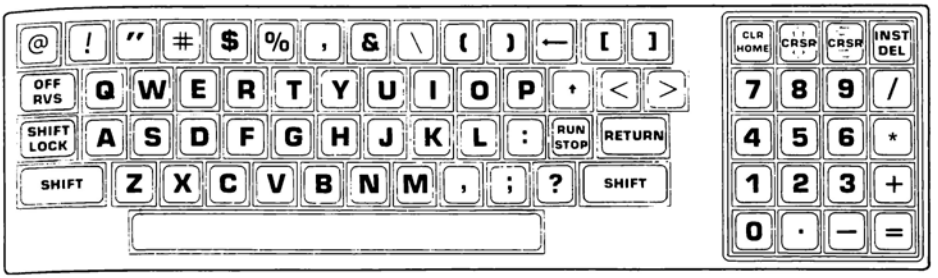
Questi tasti sono in parte presenti sulla tastiera principale e in parte su quella numerica. Sono prodotti solo con SHIFT non abbassato.

Tasti grafici



Come si è già detto in precedenza i simboli grafici sono disegnati anteriormente a tutti i tasti non di funzione o di movimento del cursore. **Sono generati in modo standard e SHIFT abbassato.**

Tasti per funzioni



Il calcolatore CBM 2001 ha la possibilità di fornire direttamente il valore di "P greco" a differenza dei calcolatori commerciali. Tale funzione è prodotta dal tasto ↑ con SHIFT abbassato.

Ricordiamo che il valore di "P greco" (π) è pari al rapporto tra una circonferenza ed il suo diametro ed è approssimativamente eguale a 3,14159265. Provate a far apparire sulla schermo questo valore battendo la riga ombreggiata e poi premendo RETURN:

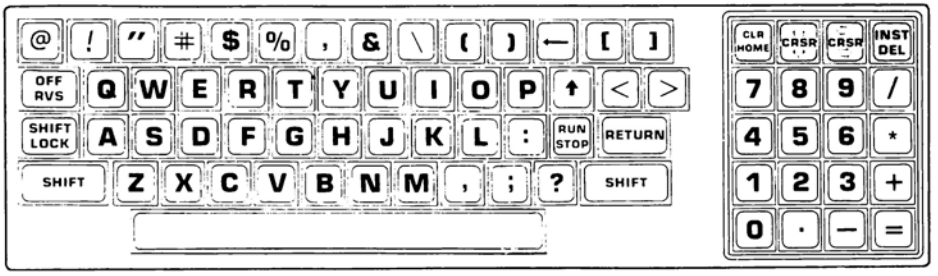
```
?π
3.14159265

READY.
```

Se π è posto tra virgolette non appare il numero 3.14159265 ma una stringa di caratteri. Infatti provate a battere:

```
?"π"
π

READY.
```

e al **premere RETURN vedrete** appunto apparire π.

Tasti di controllo del cursore



Questi tasti operano come altri già descritti precedentemente.

TASTIERA DEL CALCOLATORE PET 2001/8K

Il calcolatore PET 2001 ha una tastiera compatta, di piccole dimensioni e colorata. Essa può operare sia in modo standard che alternativo come il calcolatore PET 2001/N. Il modo standard lo si ottine con l'accensione del calcolatore, mentre per passare all'altro modo bisogna battere il comando:

```
POKE 59468,14
```

Per ritornare al modo standard:

```
POKE 59468,12
```

In Figura 1-9 è illustrata la tastiera del calcolatore PET 2001.

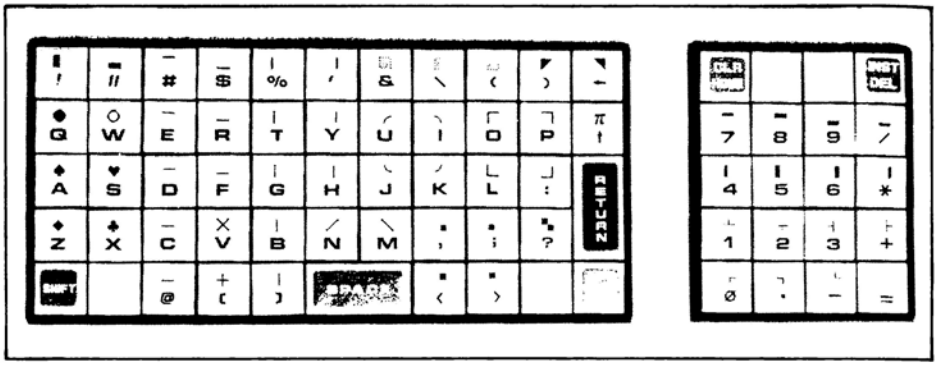
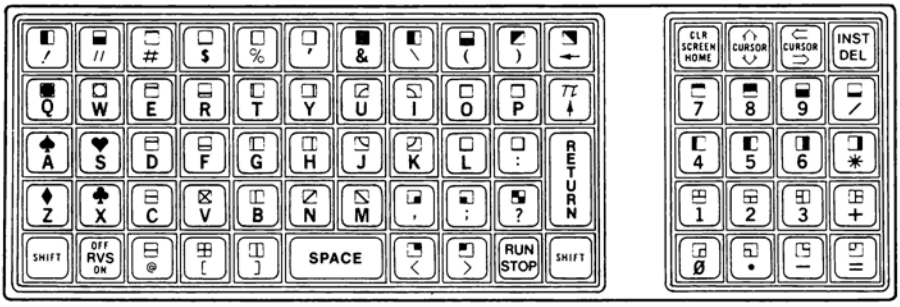


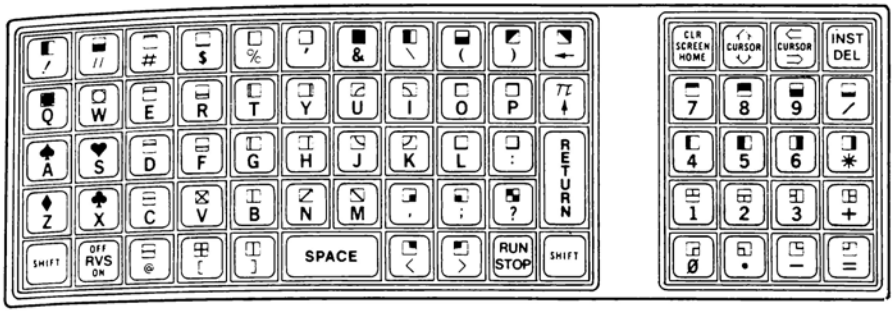
Figura 1-9: La tastiera del calcolatore PET 2001/8K

Tasti alfabetici



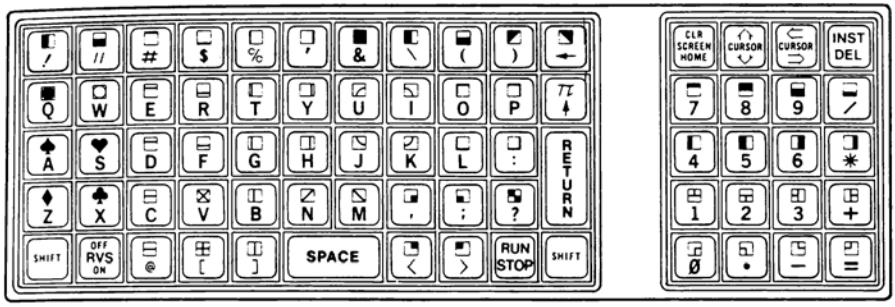
Le lettere sono generate dai tasti color argento. Nel caso di tastiera in modo standard e SHIFT non premuto si producono le lettere maiuscole; mentre nel modo alternativo si possono avere sia le lettere maiuscole che minuscole.

Tasti numerici



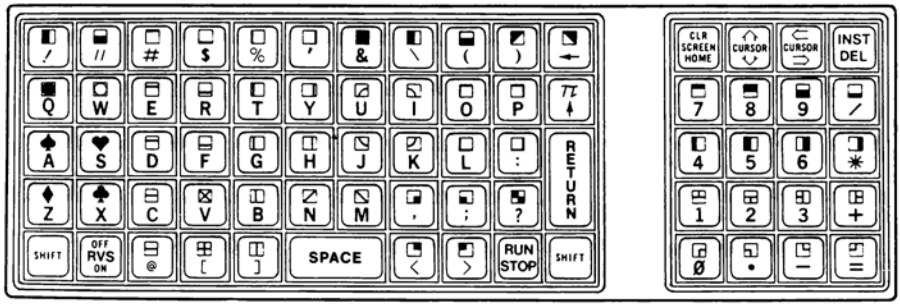
I tasti numerici sono color argento e producono le cifre con SHIFT non abbassato.

Tasti per simboli speciali



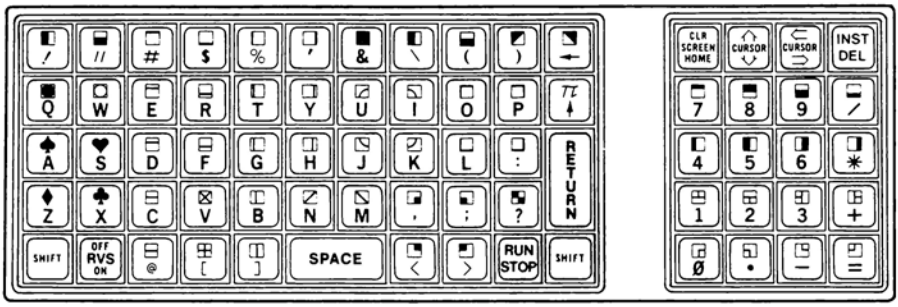
I tasti per simboli speciali, ombreggiati nella illustrazione qui sopra, sono in realtà azzurri. Essi sono raggruppati in alto e in basso, della tastiera principale, e a destra e in basso della tastiera piccola. Il tasto SHIFT non deve mai essere premuto.

Tasti grafici



I simboli grafici sono posizionati su tutti i tasti non di funzione o di movimento del cursore. **Essi sono ottenuti con tastiera in modo standard e tasto SHIFT premuto.**

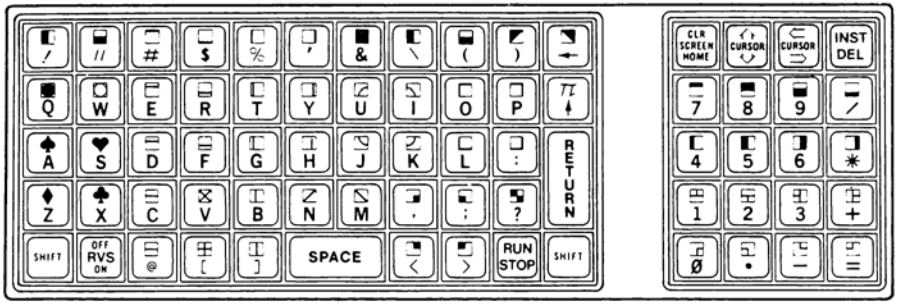
Tasti per funzioni



Questi tasti possono essere rossi o blu. **Il calcolatore PET non ha il tasto SHIFT/LOCK che è normalmente posto sopra il tasto SHIFT** di sinistra nelle altre tastiere.

Come il PET 2001/N ha il tasto per il valore "P greco" (π) posto in corrispondenza del tasto ↑.

Tasti di controllo del cursore



I quattro tasti di controllo del cursore sono di colore rosso e blu e si trovano in alto sulla tastiera piccola.

LE UNITA' A CASSETTE MAGNETICHE

Se avete un calcolatore PET esso comprende già una unità a cassette magnetiche. Se possedete invece un altro modello CBM dovete fare un collegamento esterno. In ambedue i casi i registratori a cassette sono gli stessi e permettono egualmente di memorizzare programmi o archivi di dati (in inglese "file").

Un calcolatore può collegarsi a uno o più registratori, ma solo uno sarà privilegiato e sarà visto come "primario" o "console". Nel caso del PET il registratore console è quello assemblato al suo interno (vedi Figura 1-10), mentre per gli altri

Figura 1-10: Unità a cassette interna del PET 2001

calcolatori viene riconosciuto come console il registratore connesso alla porta interfaccia J1.

Registratori a cassette esterni

Una unità a cassette esterna è illustrata in Figura 1-11. Ad uno stesso calcolatore si possono collegare contemporaneamente due registratori. Nel caso del calcolatore

Figura 1-11: Unità a cassette esterna

PET, che contiene già un registratore, la seconda unità sarà collegata all'interfaccia J3 come illustrato nelle Figura 1-5 e 1-13a. Per tutti gli altri calcolatori il primo registratore sarà collegato all'interfaccia J3. Il secondo sarà invece collegato ad un connettore interno, per i modelli 2001 (vedi Figura 1-13b), oppure come indicato nella Figura 1-13c per i modelli CBM 8000.

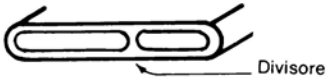
Eventuali altre periferiche, come stampanti o unità a disco saranno collegate all'interfaccia IEEE 488 senza interferire con le unità a cassette.

COLLEGAMENTO DELLE UNITA' A CASSETTE

Quando si collega un registratore esterno al calcolatore bisogna prestare molta attenzione al verso dei connettori. **Se si tentasse di forzare la connessione in modo errato si danneggerebbero i connettori stessi in maniera irreparabile.**

Il connettore maschio, posto sul calcolatore, ha infatti una fessura asimmetrica:

a cui corrisponde un divisore nel connettore femmina posto all'estremità del cavo del registratore:



Il divisore deve corrispondere quindi alla fessura così che il verso di inserimento sia univoco.

La procedura per collegare un registratore all'interfaccia esterna (come il connettore J3) è la seguente:

1. Spegnere il calcolatore.
2. Tenere il connettore, posto all'estremità del cavo in modo che il filo "blu" sia sulla destra.
3. Serrare delicatamente il connettore come indicato nella Figura 1-12. Ma attenzione, non premere assolutamente con forza.
4. Assicurarsi che la connessione tenga.
5. Riaccendere il calcolatore.

Nel caso si debba collegare invece il registratore all'interfaccia posta all'interno del calcolatore, la procedura è la seguente:

1. Spegnere il calcolatore e togliere tutti i cavi.
2. Svitare le due viti che trattengono il coperchio poste sul fondo del calcolatore.
3. Alzare il coperchio facendo attenzione però di non tirare i cavetti interni.
4. Il vostro calcolatore ha una astina per tenere sollevato il coperchio, sganciatela dal suo supporto e fissatela opportunamente al foro di una vite sul lato sinistro. Questo vi permetterà di avere ambedue le mani libere.

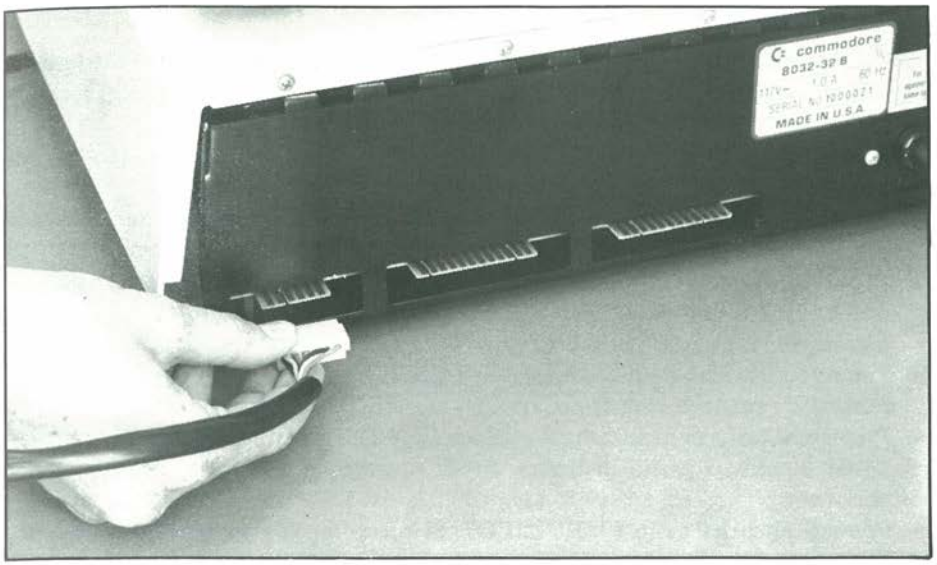


Figura 1-12: Collegamento della seconda unità a cassette esterna

5. Individuare il punto di interfacciamento sul lato sinistro della piastra a circuiti stampati come indicato in Figura 1-13b.
6. Tenere in mano il connettore, posto all'estremità del cavo del registratore, in modo che il filo "blu" sia sulla destra.
7. Serrare delicatemente il connettore come indicato in Figura 1-13b, ma non premere assolutamente con forza.
8. Assicurarsi che la connessione tenga.
9. Abbassare il coperchio e riavvitare le due viti.
10. Riattaccare i cavi e riaccendere il calcolatore.

a. Cassetta 1: PET 2001/N, CBM 2001, CBM 8000
Cassetta 2: PET 2001/8K

b. Cassetta 2: PET 2001/N, CBM 2001

c. Cassetta 2: CBM 8000

Figura 1-13: Collegamento dell'unità a cassette esterna alla piastra a circuiti stampati

Prova di funzionamento

A questo punto è necessario verificare il funzionamento meccanico delle unità a cassette e consigliamo quindi di seguire la seguente procedura di controllo sia per i tasti che per le parti meccaniche interne.

1. Accendere il calcolatore e assicurarsi che nessun tasto del registratore sia premuto ed il motorino di avanzamento del nastro sia fermo.
2. Aprire lo sportellino premendo il tasto STOP/EJECT (oppure manualmente per i vecchi modelli). Guardando all'interno del registratore premere il tasto PLAY ed osservare che le testine magnetiche si muovano. Contemporaneamente il "pinch roller" si deve spostare e il "capstan" porre in rotazione in verso antiorario. L'interno del registratore deve apparire come in Figura 1-14.
3. Premere nuovamente il tasto STOP/EJECT. Le testine magnetiche dovranno retrarsi e i perni di rotazione fermarsi.
4. Premere il tasto di avanzamento veloce FFWD. Le testine dovranno rimanere ferme mentre il perno di destra dovrà ruotare molto velocemente in verso antiorario.
5. Premere ancora il tasto STOP/EJECT. Si interrompe l'avanzamento veloce del punto 4 e il perno di destra si ferma.
6. Premere il tasto di riavvolgimento REW. Mentre le testine rimarranno ancora ferme, ruoterà invece velocemente il perno di sinistra in verso orario.
7. Per fermare il riavvolgimento premere il tasto STOP/EJECT.
8. Premere delicatamente il tasto di registrazione REC. Esso, se premuto da solo, non dovrà muoversi, ma funzionerà solo se premuto contemporaneamente al tasto PLAY e con cassetta non protetta inserita.

Se tutto funziona correttamente è possibile allora operare con il registratore. Se qualcosa non funziona verificate ancora che il calcolatore sia acceso, che per errore non si siano premuti due tasti contemporaneamente (per esempio non sia abbassato il tasto STOP) e forse anche che i tasti non siano stati premuti a sufficienza fino a sentire il "click" di scatto. Se il registratore non dovesse egualmente funzionare allora è consigliabile interpellare il Rappresentante Commodore.

Pulitura e demagnetizzazione delle testine magnetiche

Le testine magnetiche, così come sono riportate nel disegno di Figura 1-15, si possono vedere nell'interno del registratore premendo il tasto PLAY con il calcolatore spento.

Durante il normale funzionamento il nastro magnetico scorre sulle testine per cui, con il tempo, vi deposita sopra il suo stesso rivestimento di ossido. Per assicurare un corretto funzionamento del registratore è necessario quindi rimuovere periodicamente tale deposito. La pulitura sia delle due testine come anche del capstan e del pinch roller potrà esser fatta con **un fiocco di cotone bagnato in etanolo denaturato**. Prima di richiudere lasciare asciugare bene.

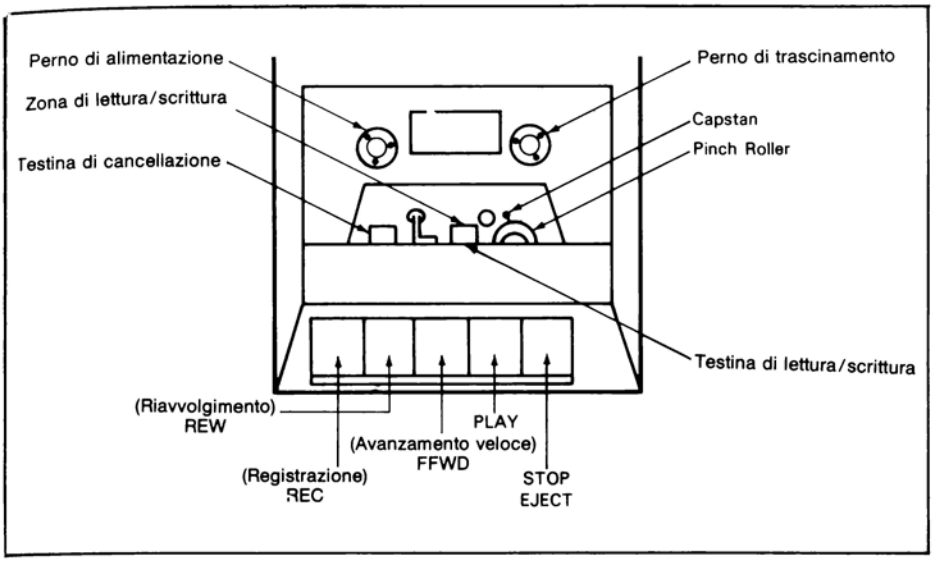


Figura 1-14: Componenti meccanici di una unità a cassette

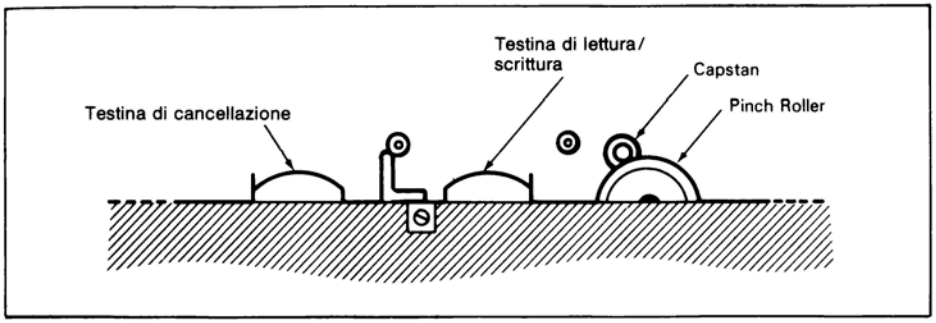


Figura 1-15: Testine magnetiche

Le testine necessitano di essere demagnetizzate periodicamente. Con l'uso prolungato le testine acquisiscono un magnetismo residuo che può influenzare notevolmente la fedeltà di registrazione e causare errori di lettura/scrittura. Per demagnetizzare le testine esistono in commercio dei dispositivi molto economici che si possono acquistare presso i negozi di fonoriproduzione.

Per demagnetizzare quindi le testine dovete spegnere il calcolatore, premere il tasto PLAY e procedere con molta attenzione come segue. Prima di accendere il demagnetizzatore dovete tenerlo almeno ad un metro di distanza dal registratore; accenderlo e avvicinarlo *lentamente* alle testine; toccare *lentamente* le testine e ogni

Figura 1-16: Un esempio di demagnetizzatore

superficie metallica che sia nelle vicinanze; *allontanarlo* quindi sempre molto lentamente e spegnerlo solo quando è tornato lontano dal registratore.

Attenzione: non avvicinare mai il demagnetizzatore a cassette già registrate! Non dimenticate infatti che il demagnetizzatore non è altro che una forte testina di cancellazione.

Più spesso effettuate registrazioni, più spesso dovrete pulire e demagnetizzare le testine. Vi consigliamo di farlo almeno una volta al mese o più sovente se riscontrate problemi di degradazione dei nastri.

Tasti di comando dell'unità a cassette

Questi tasti di comando sono posti sulla parte anteriore del registratore.

REGISTRAZIONE (REC). Il tasto REGISTRAZIONE permette il trasferimento di informazioni dalla memoria centrale alla cassetta (operazione di scrittura).

RIAVVOLGIMENTO (REW). È il tasto per il riavvolgimento veloce del nastro. Il riavvolgimento del nastro si dovrà fare molto spesso sia prima di togliere una cassetta dal registratore, sia ogni qualvolta si debbano cercare informazioni su un tratto precedente di nastro.

AVANZAMENTO VELOCE (FFWD). È il tasto per l'avanzamento veloce del nastro. Quando il registratore svolge invece funzioni di lettura/scrittura comandate dal calcolatore, esso avanza alla normale velocità di registrazione che è molto più lenta.

RIPRODUZIONE "PLAY". Questo tasto permette l'operazione di "lettura" delle informazioni dal nastro verso il calcolatore. Se premuto contemporaneamente al tasto di REGISTRAZIONE permette l'operazione di "scrittura".

STOP/EJECT. Il tasto STOP disabilita ogni altro tasto. Se un tasto non dovesse funzionare provate a premere STOP e poi nuovamente il tasto di prima.

EJECT. È il comando per l'apertura dello sportello; non era presente nelle prime versioni di registratori PET.

Inserimento e disinserimento delle cassette

Per inserire una cassetta si proceda come segue (vedere anche la Figura 1-17):

1. Premere EJECT per aprire lo sportello.
2. Tenendo in mano la cassetta come indicato in Figura inserirla nelle guide poste sotto lo sportello.
3. Chiudere lo sportello. La cassetta si collocherà così nella giusta posizione davanti alle testine.

Per estrarre la cassetta basterà sollevare lo sportello o premere il tasto EJECT, e toglierla dalle guide.

a. Drive vuoti e con il coperchio aperto

b. Modo corretto di tenere in mano una cassetta prima di inserirla

c. Drive aperto con la cassetta inserita

d. Drive chiuso con la cassetta inserita

Figura 1-17: Come inserire una cassetta

CASSETTE MAGNETICHE

Le cassette megnetiche che si utilizzano nei registratori Commodore sono come quella qui sotto riportata:

Esse possono essere acquistate sia vergini, sia contenenti programmi particolari preregistrati.

Uso delle cassette magnetiche

Qui di seguito desideriamo darvi alcuni consigli per un corretto uso delle cassette.

Quando acquistate una cassetta nuova (sia vergine che preregistrata) bilanciate la tensione del suo nastrino facendola avvolgere prima tutta in avanti (FFWD) poi tutta in dietro sino all'inizio (REW). Ciò è una precauzione per evitare possibili errori di lettura detti anche errori di "LOAD".

Non acquistate mai cassette con tempi di registrazione più lunghi di 15 o 30 minuti per evitare nastrini troppo sottili e tempi di ricerca troppo lunghi. Scegliete cassette di alta qualità, basso rumore, alta risposta e con rivestimento tipo ossido di ferro; tali cassette sono spesso denominate come "Computer Grade". Conservatele poi in un luogo fresco, asciutto e lontano da magneti e apparecchiature elettriche.

Fate molta attenzione a non toccare mai il rivestimento di ossido del nastrino!

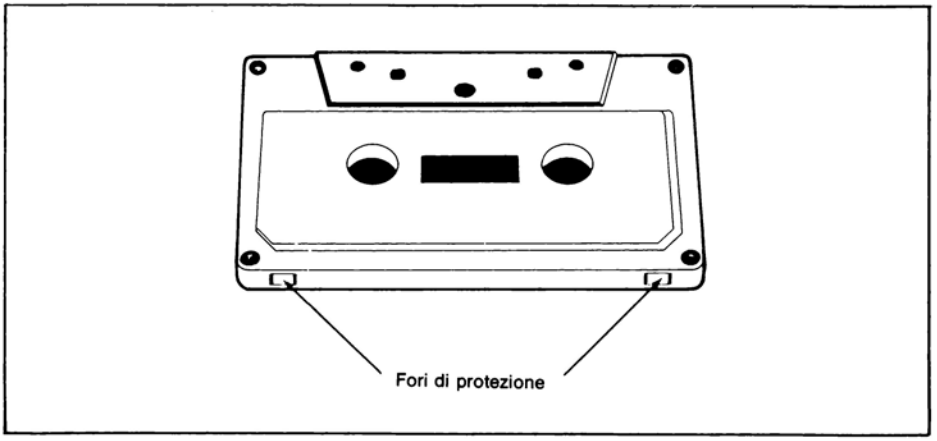


Figura 1-18: Fori di protezione contro la scrittura

Protezione da scrittura delle cassette

Se abbiamo una cassetta con delle registrazioni importanti e desideriamo proteggerla, così da poter fare delle operazioni di sola lettura e mai di scrittura, **possiamo interdire la scrittura aprendo i due fori posti sul suo fondo** (vedi Figura 1-18). I fori sono due perchè corrispondono alle due tracce. Se desiderate quindi proteggere una sola traccia della vostra cassetta strappate la chiusura del foro corrispondente.

Se in futuro vorrete egualmente scrivere su una cassetta protetta basterà coprire con del normale nastro adesivo il foro corrispondente.

UNITA' A DISCO CBM

Nella linea di prodotti Commodore sono presenti anche unità periferiche a disco del tipo a dischetto flessibile "floppy". I modelli sono il CBM 8050 e CBM 2040 ambedue a doppio "drive" (drive è il supporto meccanico per l'avanzamento e/o rotazione di un nastro o un disco). Il floppy è un dischetto flessibile le cui dimensioni ricordano un disco musicale a 45 giri.

Le unità CBM 8050 e 2040 sono presentate nelle Figura 1-19 e 1-20. Il modello CBM 2040 memorizza circa 171.500 byte per dischetto con la tecnica detta a "singola densità".

Il modello CBM 8050 è invece a "doppia densità" e memorizza circa 522.000 byte su ogni dischetto. **Le specifiche delle due unità a disco sono riportate nelle Tabelle 1-2 e 1-3.**

Tabella 1-2: Specifiche dell'unità a disco CBM 8050

Fisiche		**Elettriche**	
Dimensioni:		Alimentazione:	
altezza	6.5"	Tensione	vedere retro macchina
larghezza	15"	Frequenza	vedere retro macchina
profondità	14.35"	Potenza	50 Watt
Circuiti Integrati		**Drive:**	
Controller:		Shugart	SA 390 (2)
6502	Microprocessore	Dischetti	Mini 5, 1/4" standard
6530	I/O, RAM, ROM		
6522	I/O, Timer.		
Interfacce:		**Memoria dischetti:**	
6502	Microprocessore	Capacità totale	533.248 byte
		Sequenziale	521.208 byte
6532 (2)	I/O, RAM, Timer	Random	da 464.312 a 517.398 byte a seconda del file
6564 (2)	ROM	Settori per Traccia	da 23 a 29
		Byte per Settore	256
Altri:		Tracce	77
6114 (8)	4 x 1K RAM	Blocchi	2083

Tabella 1-3: Specifiche dell'unità a disco CBM 2040

Fisiche		**Elettriche**	
Dimensioni:		Alimentazione:	
altezza	6.5"	Tensione	vedere retro macchina
larghezza	15"	Frequenza	vedere retro macchina
profondità	14.35"	Potenza	50 Watt
Circuiti Integrati		**Drive:**	
Controller:		Shugart	SA 390 (2)
6504	Microprocessore	Dischetti	Mini 5, 1/4" standard
6530	I/O, RAM, ROM		
6522	I/O, Timer		
Interfacce:		**Memoria dischetti:**	
6502	Microprocessore	Capacità totale	176.640 byte
6532 (2)	I/O, RAM, Timer	Sequenziale	170.180 byte
6564 (2)	ROM	Random	170.850 byte
		Settori per Traccia	da 17 a 21
		Byte per Settore	256
Altri:		Tracce	35
6114 (8)	4 x 1K RAM	Blocchi	690

Collegamento delle unità a disco al calcolatore CBM

Per collegare una unità a disco al calcolatore CBM, si proceda come segue:

1. Disinserire il cavo di alimentazione del calcolatore.
2. Mediante un cavo di connessione cón connettori tipo IEEE/CBM, inserire il connettore IEEE nella presa, posta sul retro in alto a sinistra, dell'unità a disco (vedere Figura 1-21). Bloccare quindi il connettore girando le sue due viti in

Figura 1-19: Unità a doppio disco floppy CBM 8050

Figura 1-20: Unità a doppio disco floppy CBM 2040

3. Inserire delicatamente l'altro connettore nel punto di connessione J1 (vedi Figura 1-5) sul retro del calcolatore CBM. Fare attenzione che la scritta "Commodore" su questo connettore appaia sul lato superiore. Il collegamento nel suo insieme deve apparire come in Figura 1-21.

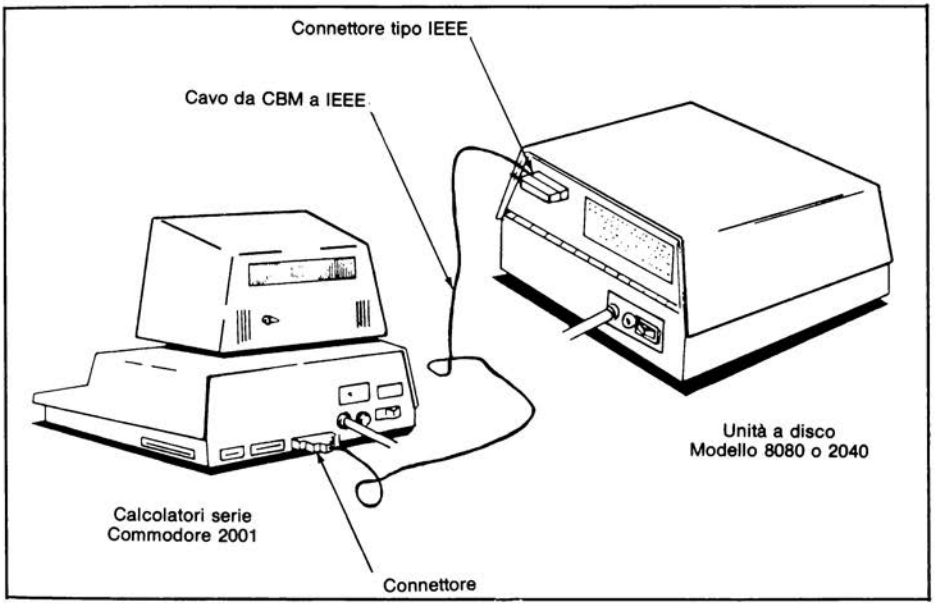


Figura 1-21: Collegamento tra calcolatore e unità a disco

4. Inserire il cavo di alimentazione del disco in una presa di corrente facendo molta attenzione che la tensione sia quella prescritta.
5. Inserire nuovamente il cavo di alimentazione del calcolatore.

Eseguire quindi la seguente prova di accensione.

Prova di accensione

1. Accendere il calcolatore ed assicurarsi che funzioni correttamente.
2. Solo per il drive modello 2040: aprire i due sportellini tirando delicatamente con l'indice. Per il modello 8050 non è necessario.
3. Assicurarsi che i drive non contengano dischetti.
4. Accendere il drive mediante l'interruttore posto sul retro a sinistra.

Modello 8050: i tre indicatori luminosi verdi, posti sul frontale, devono lampeggiare due volte e poi quello centrale deve rimanere acceso, mentre gli altri due devono spegnersi.

Modello 2040: i tre indicatori rossi devono lampeggiare e poi spegnersi. Attenzione: alcuni drive fanno un leggere ronzio durante l'accensione.

Se alcune luci rimangono accese spegnete e riaccendete dopo qualche minuto. Se qualche cosa continuasse a rimanere irregolare chiamate il Rappresentante Commodore.

Indicatori luminosi

Ambedue i modelli di unità a disco hanno tre indicatori luminosi posti sul frontale (vedi Figura 1-22). Essi non sono a lampadina, ma a LED (Light Emitting Diode). Due sono posti sul drive 0 e drive 1 e indicano se accesi che l'unità è in funzionamento. Il terzo è un rivelatore di errore.

Il modello 8050 ha tre LED *verdi*. Come già detto quelli sui drive, se accesi, indicano lo stato di funzionamento; il terzo indica l'accensione dell'unità. **Quest'ultimo rivela anche la presenza di errori cambiando da verde in rosso e rimanendo acceso sino a che non si corregge l'errore.**

Il modello 2040 ha tre LED *rossi*, due per indicare lo stato di funzionamento dei drive, mentre il terzo è solo indicatore di errore.

Inserimento ed estrazione dei dischetti

Ogni dischetto vero e proprio è contenuto in due buste. La prima è una busta di carta che ha lo scopo di conservare e proteggere il dischetto quando non è inserito nel drive. **La seconda invece è una vera e propria cartuccia che non bisogna mai tentare di aprire** (vedi Figura 1-23). Il dischetto deve essere inserito nel drive con tutta questa seconda busta: essa ha lo scopo sia di proteggere il dischetto che di mantenerlo pulito.

Figura 1-22: Indicatori luminosi a LED dell'unità a disco CBM 2040

Rammentate inoltre che **i calcolatori CBM impiegano dischetti, il cui diametro è di 5-1/4 pollici e hanno la ripartizione dei settori fatta a software.** Se avete il dubbio che un vostro dischetto abbia o meno i settori ripartiti a software fate la seguente prova (vedi Figura 1-24):

1. Prendete in mano il dischetto, ovviamente con la sua cartuccia, come indicato nella Figura 1-24 al punto 1.
2. Inserite due dita nel foro centrale.

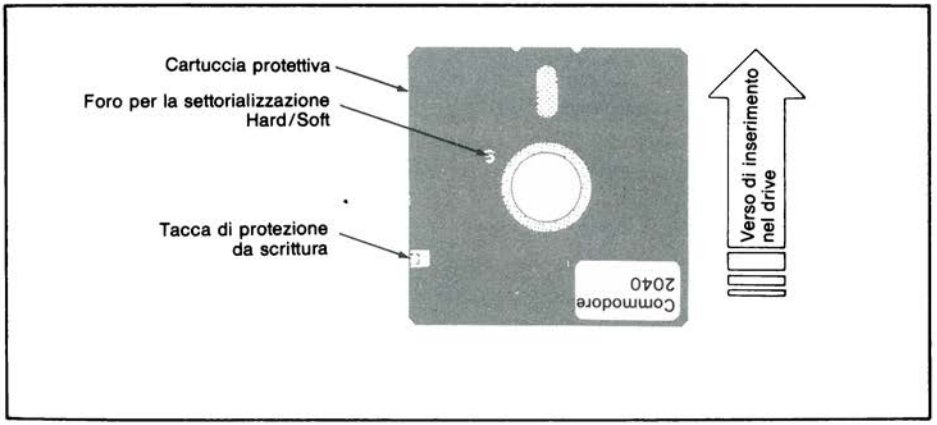


Figura 1-23: Dischetto Floppy.

a. Punto 1 b. Punto 2 c. Punto 3, 4

Figura 1-24: Come stabilire il tipo di settorializzazione

3. Individuate il piccolo foro nella cartuccia vicino al foro centrale.
4. Ruotate con le due dita il dischetto fino a che troverete un piccolo foro in corrispondenza a quello della cartuccia.

Se trovate un solo foro i settori sono ripartiti a software diversamente, cioè con più fori, la ripartizione è ad hardware e il dischetto non può essere utilizzato con il vostro calcolatore.

Inserimento nel modello 8050

1. *Non toccate mai la superficie magnetica del dischetto potreste danneggiarla irreparabilmente.* Tenete quindi il dischetto per la sua cartuccia e inseritelo nel drive

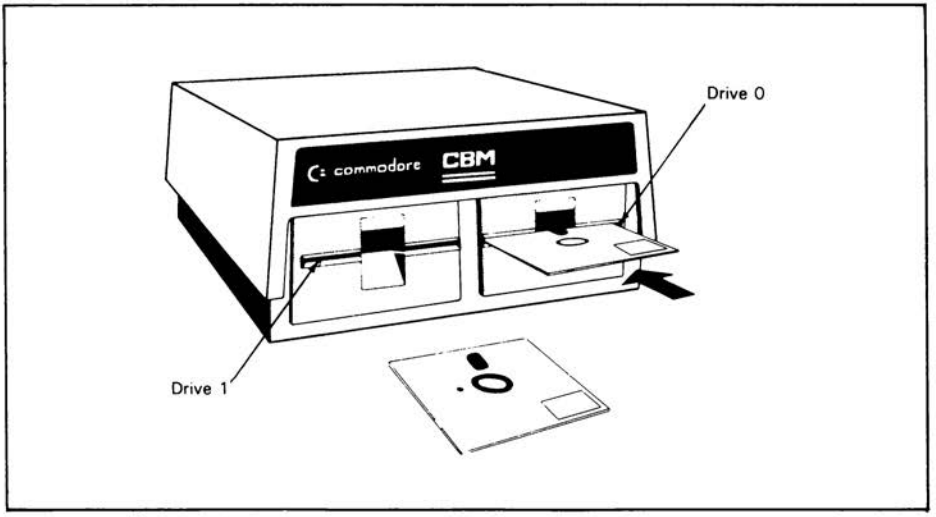


Figura 1-25: Inserimento del dischetto nell'unità CBM 8050

a. Punto 1

b. Punto 2

Figura 1-26: Estrazione del dischetto dall'unità CBM 8050

nel verso indicato in Figura 1-23. Fate attenzione che la tacca di protezione da scrittura sia sulla sinistra.

2. Come indicato in Figura 1-25 spingete delicatamente il dischetto fino a che non sentite un "click". Se non riuscite subito riprovate, *ma non forzate mai potreste danneggiare sia il dischetto che il drive.*
3. Chiudete quindi il portellino spingendo la sua maniglia verso il basso fino allo scatto.

Estrazione dal modello 8050

Non estraete mai un dischetto dal suo drive se il corrispondente indicatore LED è acceso.

1. La maniglia del portello del drive dovrebbe già essere in posizione abbassata. Con due dita fate una leggera pressione sulla maniglia verso l'alto: il dischetto sarà quindi automaticamente sospinto in fuori (vedi Figura 1-26).
2. Estraete quindi totalmente il dischetto cercando di *non toccare la superficie magnetica nè di piegarlo.*
3. Inseritelo subito nella sua busta di protezione.

Inserimento nel modello 2040

1. Tenete il dischetto per la sua cartuccia *cercando assolutamente di non toccare la superficie magnetica*, con la sua faccia principale verso l'alto e la tacca di protezione da scrittura sulla sinistra.
2. Tirate delicatamente il portellino con il dito indice sino a che esso non si apra (Figura 1-27).
3. Con molta delicatezza spingete il dischetto nel drive fino a sentire uno scatto. Se non entra dolcemente estraetelo subito e riprovate nuovamente, *ma senza forzare.*
4. Chiudete il portellino spingendolo verso il basso.

Figura 1-27: Apertura del portello del drive (CBM 2040)

Estrazione dal modello 2040

Non togliete mai un dischetto se il corrispondente indicatore di LED è acceso.

1. Aprite il portellino tirandolo con il dito indice come indicato nella Figura 1-27.
2. Estraete quindi il dischetto e mettetelo subito nella sua busta.
3. Chiudete il portellino premendolo verso il basso fino allo scatto.

DISCHETTI "FLOPPY"

Conservazione dei dischetti

I dischetti devono essere trattati con molta cura perchè se danneggiati non possono più essere letti e quindi l'informazione che memorizzavano andrà completamente persa.

1. Appena estraete un dischetto dal drive ponetelo subito nella sua busta.
2. *Non tentate mai di estrarlo dalla sua cartuccia di plastica.*
3. Non scrivete mai sull'etichetta del dischetto salvo che con un pennarello. È preferibile scrivere sulle etichette separatamente e poi incollarle sopra.
4. Non toccate mai la superficie magnetica del dischetto ne tanto meno tentate di pulirla.

5. Evitate di fumare vicino alle unità a disco. Residui di cenere o tabacco possono graffiare il dischetto.
6. **Teneteli assolutamente lontano da campi magnetici.**
7. Non esponeteli alla luce o al calore intensi.

Protezione da scrittura

L'operazione di scrittura su un dischetto può essere inibita coprendo la apposita tacca di "protezione da scrittura" (vedi Figura 1-23). Solo se questa tacca è aperta si può scrivere sul dischetto; se invece è coperta non si può scrivere. Per coprirla si può usare del normale nastro adesivo, ma è possibile acquistarne uno più adatto allo scopo.

Figura 1-28: Stampante modello CBM 2022

Figura 1-29: Stampante modello CBM 2023

LE STAMPANTI CBM

La CBM offre due modelli di stampanti che si differenziano per il sistema di trascinamento della carta: **il modello CBM 2022 "Tractor Feed", che impiega la carta a modulo continuo con avanzamento a "sprocket" e il modello CBM 2023 "Matrix or Friction Feed" che impiega invece carta comune con avanzamento a frizione.** I due modelli sono presentati nelle Figure 1-28 e 1-29.

Per quanto concerne le altre caratteristiche le due stampanti sono eguali: formano i caratteri mediante una matrice di 7x6 punti (in maniera analoga a quanto avviene nei display video); stampano 80 caratteri per linea alla velocità di una linea al secondo (60 linee al minuto); impiegano nastro per macchina da scrivere nero e rosso con occhielli per invertirne la direzione. **Nella Tavola 1-4 sono riportate tutte le specifiche delle stampanti.**

Collegamento tra stampante e calcolatore CBM

Il collegamento di una stampante al calcolatore si effettua nel modo seguente:

1. Staccate il calcolatore dalla presa di corrente.
2. Nel caso di collegamento *diretto* della stampante al calcolatore seguite i seguenti punti:
   a. Prendete un cavo di connessione CBM/IEEE e applicate il connettore IEEE alla presa posta sul retro a destra della stampante (come indicato nella Figura 1-30). Girate le due viti in verso orario per bloccare il connettore IEEE.
   b. Applicate l'altro connettore al punto J1 del calcolatore prestando attenzione che la scritta "Commodore" appaia verso l'alto.
3. Nel caso di collegamento *multiplo* di più periferiche alla stessa interfaccia J1 del calcolatore, che ricordiamo è unica, è necessario collegare in cascata il calcolatore, l'unità a disco e la stampante:
   a. Collegate prima l'unità a disco al calcolatore come già indicato nella Figura 1-21.
   b. Mediante un cavo di connessione tipo IEEE/IEEE collegate la stampante all'unità a disco ricordandovi sempre di bloccare i connettori mediante le loro viti. Sul retro dell'unità a disco si avranno così sovrapposti i connettori dei due cavi provenienti dal calcolatore e dalla stampante. Questa tecnica di collegamento viene detta a "festoni"; i dati verso la stampante passano attraverso i due cavi.
4. Ricollegate il calcolatore ad una presa di corrente.
5. Collegate la stampante ad una presa di corrente.
6. Accendete il calcolatore ed assicuratevi che funzioni correttamente.
7. Accendete la stampante (il suo interruttore è posto su un lato). La testina di scrittura dovrà subito portarsi a destra e poi a sinistra.

Tabella 1-4: Specifiche delle stampanti CBM 2022 e 2023

Specifiche della stampante modello 2022

Metodo di stampa	A impatto con matrice a punti seriale
Velocità	70 linee/min o 150 caratteri/sec. (massimo)
Direzione di stampa	Unidirezionale
Numero di colonne	80
Ampiezza caratteri	6x7
Dimensione caratteri	Altezza 0,11", larghezza 0.10"
Spazio tra le colonne	1/10" (10 caratteri per pollice)
Spazio tra le linee	Programmabile
Copie	3 compreso l'originale
Nastro	Di nylon con occhielli
Durata del nastro	2 milioni di caratteri
Larghezza carta	10" (carta per calcolatori pieghevole)
Formato fori trascinamento	8,5 + 0,5 x 2 (margini sprocket) Distanza tra i fori: 0,5" longitudinale 9,0" laterale 5/32" diametro

Specifiche della stampante modello 2023

Eguali a quelle del modello 2022 salvo:

Spazio fra le linee	1/6" (6 linee per pollice)
Formato fori trascinamento	Non applicabile l'avanzamento con sprocket

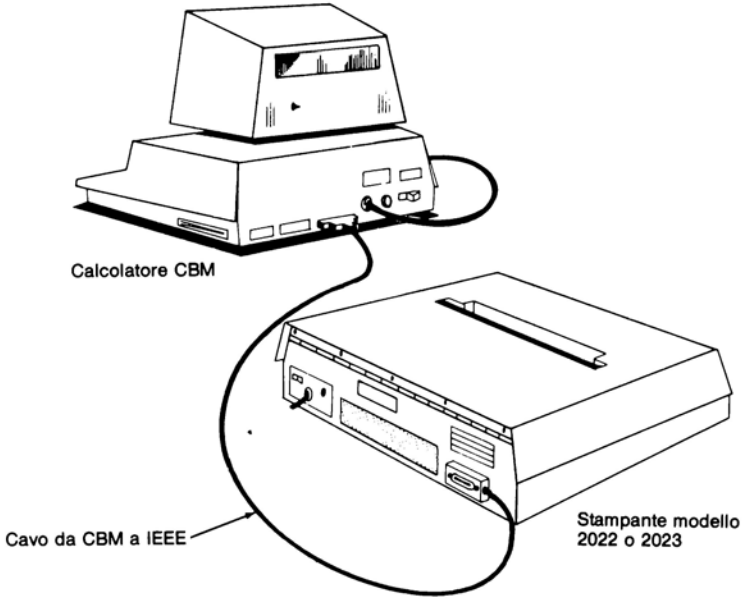


Figura 1-30: Collegamento tra stampante e calcolatore

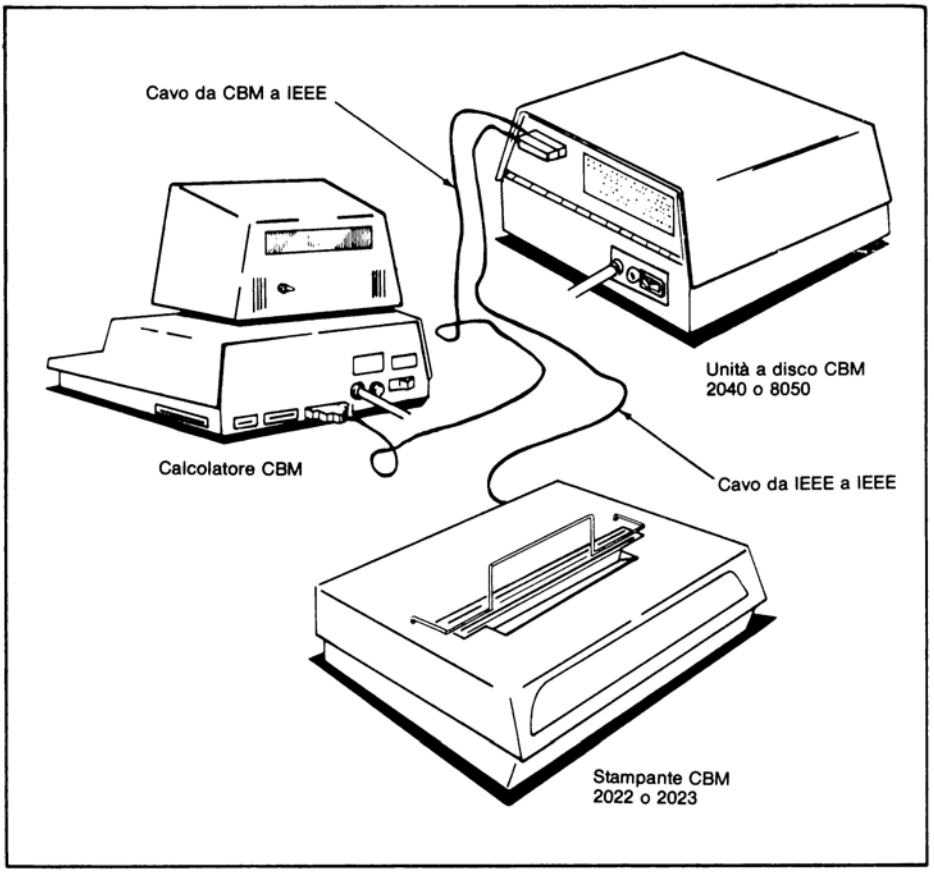


Figura 1-31: Collegamento multiplo tra stampante, unità a disco e calcolatore

Se la testina non dovesse muoversi provate a spegnere, riaccendere e controllare ancora le connessioni; se il difetto permane allora è preferibile chiamare il Rappresentante Commodore.

Come installare il nastro

1. Sollevate il coperchio della stampante.
2. Tenete nella mano sinistra il rocchetto su cui è avvolto tutto il nastro e nella destra quello vuoto ove appare l'occhiello di inversione del nastro (vedi Figura 1-32).
3. Fissate il rocchetto di destra sul suo supporto (posizione 1 nella Figura 1-33).

4. Svolgete un pò di nastro e fategli seguire il percorso 2, 3 e 4 assicurandovi che l'occhiello sia tra i punti 2 e 3. Continuate a svolgere il nastro attraverso i punti 5 e 6 facendo sempre molta attenzione di non attorcigliare il nastro.
5. Fate passare il nastro dietro la testina di scrittura e poi attraverso i punti 7, 8, 9, 10 e 11: attenzione sempre a non attorcigliarlo!
6. Riavvolgete il nastro così che sia un pò in tensione e bloccate il rocchetto di sinistra sul suo supporto (posizione 12).
7. Chiudete infine il coperchio.

Come caricare la carta

Ogni stampante richiede un metodo diverso a seconda del tipo di avanzamento meccanico della carta.

Caricamento nella stampante CBM 2022

Questa stampante impiega carta standard del tipo a modulo continuo con fori di avanzamento sui lati. La massima larghezza della carta può essere di 10 pollici (vedi Figura 1-34).

Per caricare la carta procedete come segue:

1. Con ambedue le mani tirate delicatamente in avanti il dispositivo di trascinamento a trattori.
2. Spingete delicatamente la carta dal retro verso l'avanti come indicato in Figura 1-35.
3. Quando la carta è davanti spingete in dietro tutto il dispositivo di trascinamento.
4. Aprite i due ferma-carta davanti ai trattori.
5. Fate combaciare i fori della carta con i denti dei trattori. Se non vi è possibile, fate la seguente regolazione:
   a. Sollevate la leva vicino ai trattori verso di voi.
   b. Aggiustate la posizione dei trattori in modo che i denti di avanzamento collimino con i fori della carta.
   c. Riportate la leva nella posizione di blocco.
6. Quando la carta è ben sistemata davanti ai trattori chiudete i due ferma-carta.
7. Accendete la stampante (l'interruttore è posto in basso a destra sul lato destro).
8. La carta può essere fatta avanzare in due modi:
   a. Mediante l'interruttore posto in basso a sinistra sul fronte della stampante.
   b. Oppure manualmente mediante le due manopole poste ai lati del dispositivo di avanzamento della carta.

Figura 1-32: Applicazione del nastro

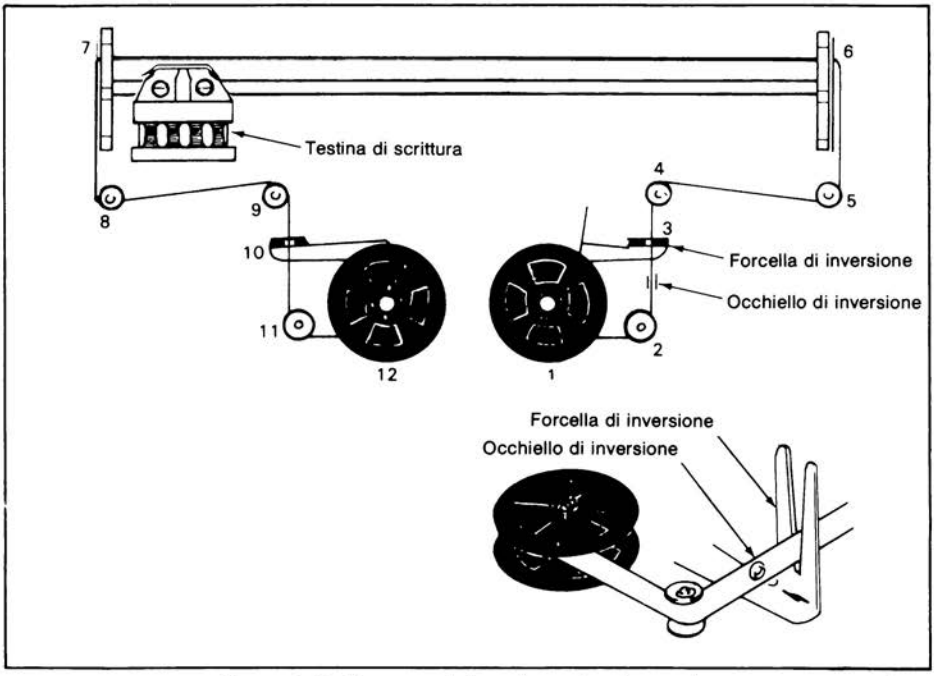


Figura 1-33: Percorso del nastro nella stampante

Figura 1-34: Stampante modello CBM 2022

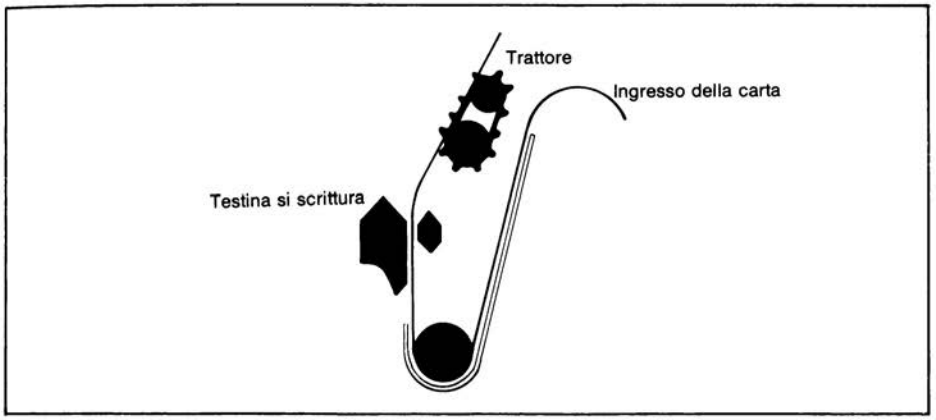


Figura 1-35: Percorso della carta nella stampante CBM 2022

Caricamento nella stampante 2023

Questa stampante può scrivere sia su fogli singoli che su carta ripiegata tipo modulo continuo, ma senza fori. Anche in questo caso la massima larghezza della carta è di 10 pollici.

Per caricare la carta procedete come segue:

1. Accendete la stampante (l'interruttore è sul retro a destra).
2. Inserite l'inizio della carta (come indicato in Figura 1-36) poi, mentre la spingete ancora, premete il tasto di avanzamento automatico posto in basso a sinistra.

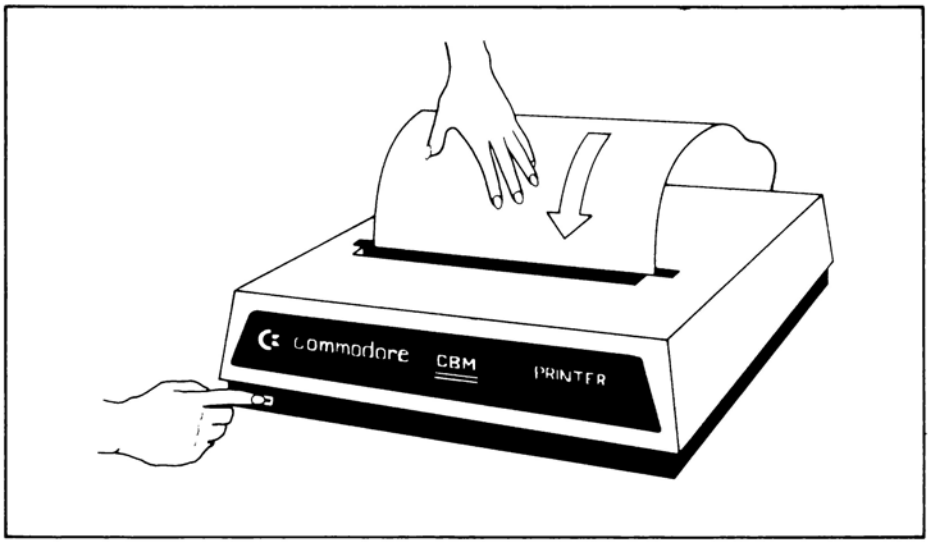


Figura 1-36: Inserimento della carta nella stampante CBM 2023

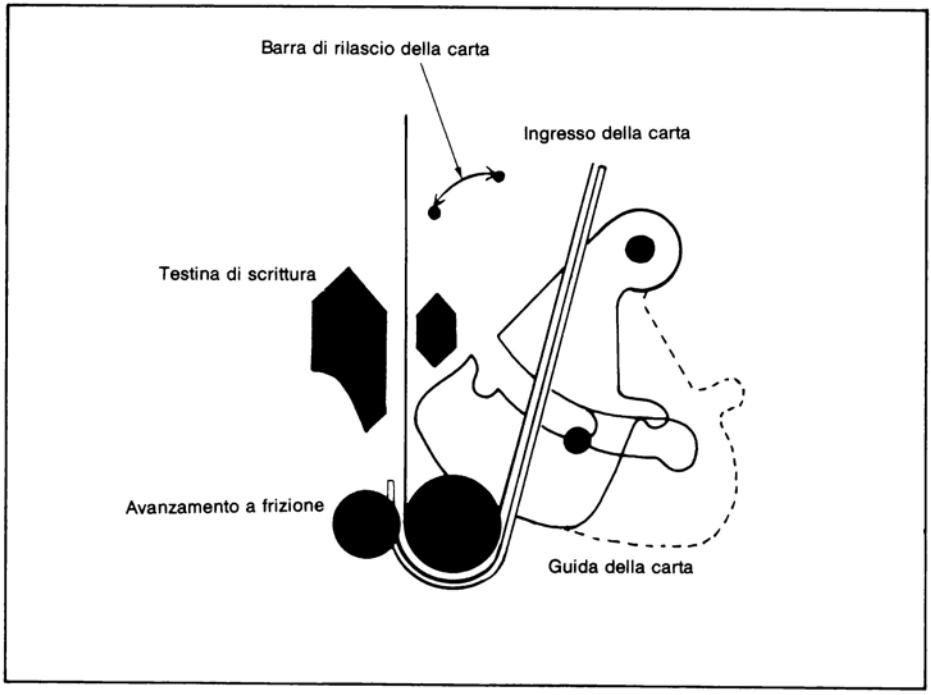


Figura 1-37: Percorso della carta nella stampante CBM 2023

```
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←  !"#$%&'()*+,-.
/0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←  !"#$%&'()*+,-./0123456789:;<=
>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKL
MNOPQRSTUVWXYZ[\]↑←  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[
\]↑←  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←  !"#$%&'()
*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←  !"#$%&'()*+,-./012345678
9:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFG
HIJKLMNOPQRSTUVWXYZ[\]↑←  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUV
WXYZ[\]↑←  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←  !"#$
%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←  !"#$%&'()*+,-./0123
456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←  !"#$%&'()*+,-./0123456789:;<=>?@AB
CDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQ
RSTUVWXYZ[\]↑←  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
```

Figura 1-38: Prova della testina di scrittura

3. Se desiderate aggiustare la posizione della carta o anche toglierla, dovete muovere la leva di rilascio della carta come si opera con una comune macchina da scrivere (vedi Figura 1-37).

Prova della testina di scrittura

Dopo avere caricato la carta e installato il nastro, dovete ancora provare la testina per essere sicuri che la stampante funzioni correttamente. **Attenzione, non fate mai questa prova ne usate la stampante se prima non avete caricato la carta perchè danneggereste sicuramente la testina!**

Procedete allora come segue:

1. Spegnete la stampante.
2. *Mentre tenete premuto il tasto di avanzamento carta* riaccendete la stampante. Subito la macchina batterà una lunga serie di caratteri di prova: numeri, lettere e simboli speciali, ma non caratteri grafici. Un esempio è indicato in Figura 1-38.

Se notate qualche difetto chiamate il Rappresentante Commodore, diversamente potete iniziare il vostro lavoro.

CAPITOLO 2

IMPIEGO DEI CALCOLATORI CBM

In questo capitolo descriviamo come operare, attraverso la tastiera, con un calcolatore CBM collegato ad unità a disco, a cassette e a stampanti.

Un calcolatore è una macchina che esegue delle istruzioni (in inglese "statements") **che fanno parte di un procedimento logico pensato dall'operatore.** In particolare i calcolatori CBM "capiscono" istruzioni scritte nel linguaggio CBM BASIC (Begining All-purpose Symbolic Instruction Code).

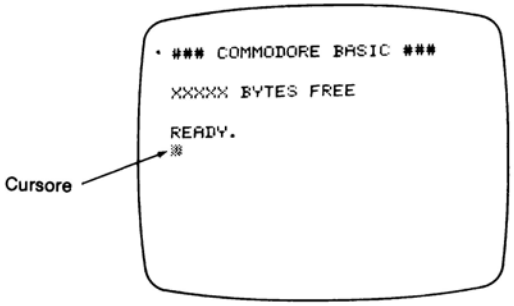
Le istruzioni BASIC possono essere lunghe al massimo 80 caratteri. Se avete un calcolatore della serie 8000 questi 80 caratteri possono stare tutti su una linea del display, mentre con gli altri calcolatori occuperanno due righe di 40 caratteri ciascuna. **Ogni linea di istruzione deve essere terminata battendo il tasto RETURN.** Se battete una istruzione più lunga di 40 caratteri, nel caso di display piccolo, il cursore andrà automaticamente a capo. Se per errore battete una istruzione più lunga di 80 caratteri il calcolatore vi risponderà ?SYNTAX ERROR appena voi battete il tasto RETURN.

Per terminare una singola linea di istruzione e avvisare quindi il calcolatore di riceverla e analizzarla dovete sempre battere RETURN.

Le istruzioni BASIC possono essere caricate in due modi: nel "modo immediato" e nel "modo differito" (o "a programma"). Nel primo caso esse sono eseguite immediatamente, sono generalmente corte e non vengono memorizzate nella memoria centrale. Nel secondo caso invece sono memorizzate e verranno eseguite solo quando verrà dato uno speciale comando al calcolatore.

MODO IMMEDIATO

Appena acceso un calcolatore CBM funziona in modo immediato e questo viene segnalato dalla presenza del cursore luminoso sullo schermo.



Il calcolatore lavora sempre in modo immediato sino a che non viene eseguito un programma BASIC. Al termine dell'esecuzione ritorna in modo immediato.

LA TASTIERA IN MODO IMMEDIATO

Il cursore si posiziona sullo schermo nel punto dove apparirà il prossimo carattere battuto alla tastiera. Ogni carattere battuto appare sempre sullo schermo così che il cursore si sposta passo passo a destra in attesa del prossimo carattere (vedi Figura 2-1). Ovviamente alcuni tasti non fanno apparire alcun simbolo come nel caso della spaziatura. Battendo il tasto CURSOR si sposta in corrispondenza il cursore a destra/sinistra, in alto/basso. Con RETURN il cursore va a posizionarsi a sinistra sulla riga successiva.

In modo immediato ogni sequenza di caratteri che battete sulla tastiera viene subito interpretata come istruzione BASIC. Quindi se non rispettate la sintassi del linguag-

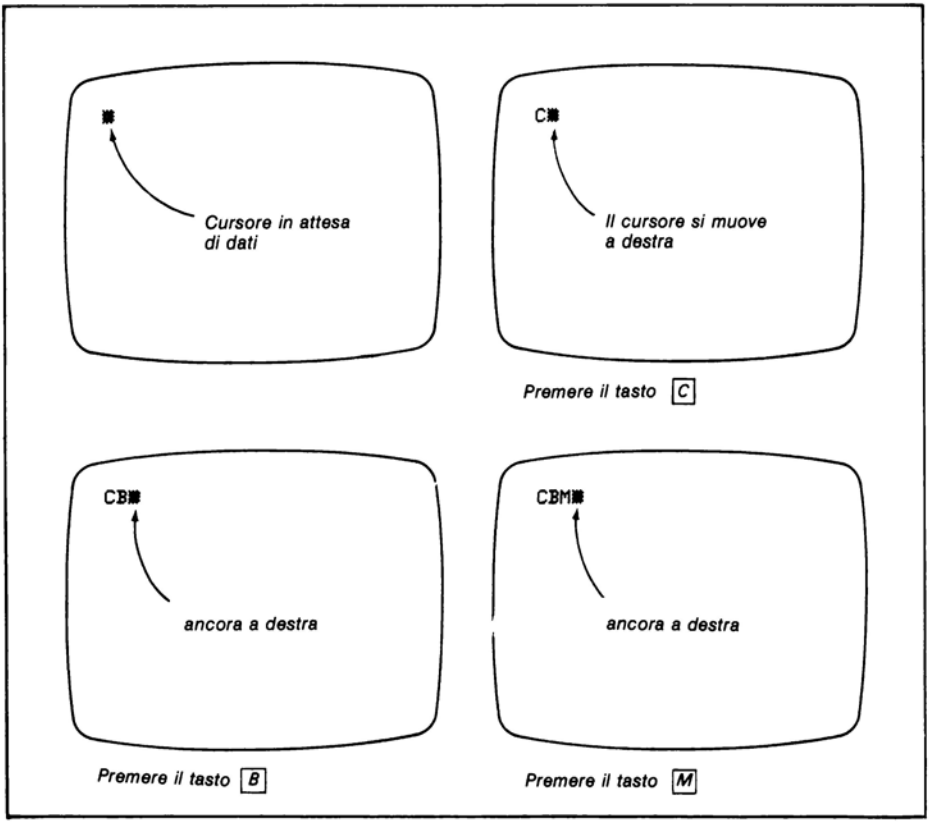


Figura 2-1: Ingresso di dati caricati dalla tastiera in modo immediato

gio BASIC vi sarà segnalato errore.

In Figura 2-2 sono illustrate alcune istruzioni valide per il modo immediato. Appena il tasto RETURN è battuto, l'istruzione viene eseguita e fornisce gli eventuali risultati.

In Figura 2-3 sono battuti caratteri alfanumerici incomprensibili al BASIC così che viene dato l'errore ?SYNTAX ERROR. Se battete però solo caratteri grafici, come nell'esempio 3, il sistema BASIC non tenta di tradurli e non da errore. Questo vi permette di disegnare liberamente sullo schermo.

Istruzioni in modo immediato

Le istruzioni che entrano in modo immediato non devono assolutamente iniziare con un numero. Successivamente esamineremo quali sono le istruzioni che si possono dare in modo immediato, ma per ora ci limitiamo a descrivere alcune istruzioni di stampa, di calcolo aritmetico e di movimento del cursore.

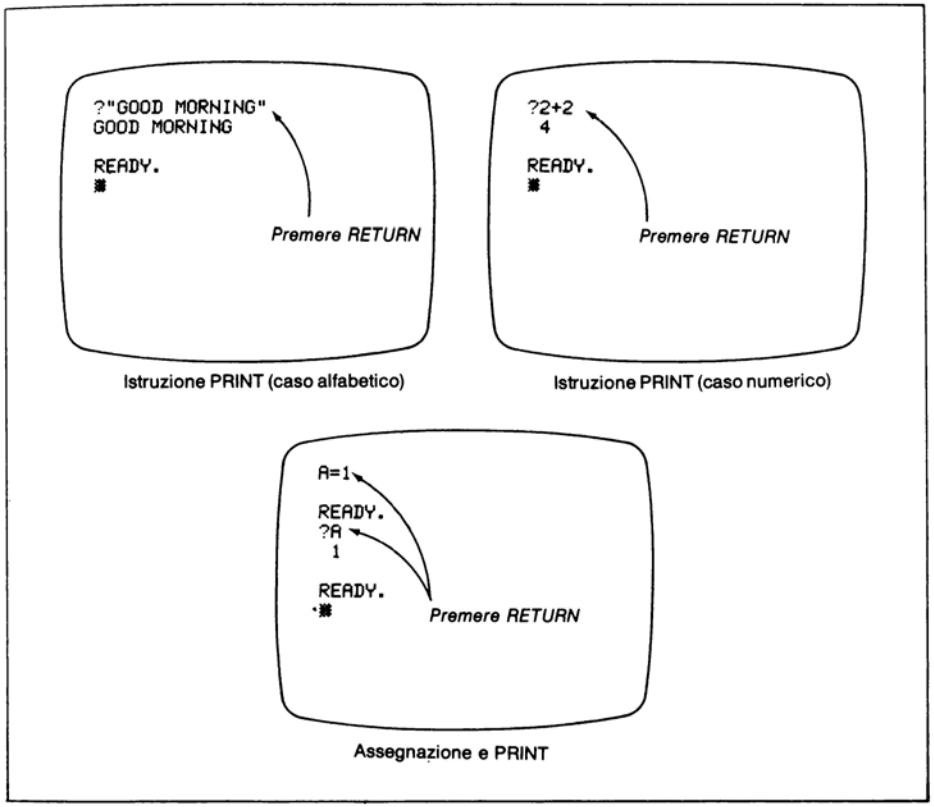


Figura 2-2: Esempi di istruzioni caricate in modo immediato

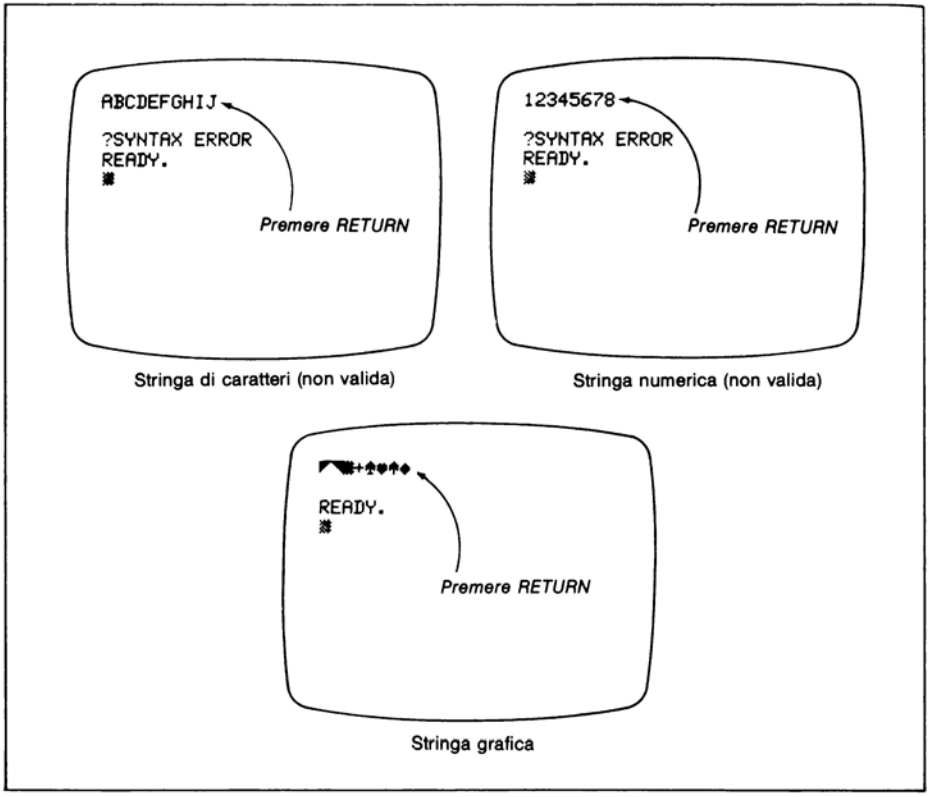


Figura 2-3: Esempi di dati (non istruzioni) caricati in modo immediato

L'istruzione PRINT

$$\left\{ \begin{matrix} \text{PRINT} \\ ? \end{matrix} \right\} \text{dati}$$

L'istruzione PRINT, che è la più usata in modo immediato, permette di rappresentare dati sullo schermo. Subito dopo il comando RETURN appaiono sullo schermo, alla riga successiva, i dati elencati nell'istruzione PRINT.

Invece della parola PRINT si può usare la forma abbreviata "?".

Stampa di stringhe

Una **"stringa" è una sequenza di caratteri alfanumerici racchiusa tra virgolette**; ecco, alcuni esempi:

```
"HAI!"
"SINERGIA"
"1234567"
"IL TOTALE È 34,54"
"MILANO, VIA CARCANO 63"
```

In una stringa possono essere inseriti caratteri alfabetici, numerici, simboli speciali e grafici, comandi del cursore e il comando REVERSE ON/OFF. Gli unici caratteri che non possono essere inseriti in una stringa sono i comandi RUN/STOP e RETURN.

Una stringa può essere visualizzata direttamente con un comando PRINT, ma anche associandola ad una opportuna "variabile di stringa" e poi dando l'istruzione di stampare la variabile.

Per visualizzare una stringa battete PRINT (o la sua abbreviazione "?") e la stringa tra virgolette.

```
        PRINT "stringa"
dove:
        stringa è il testo di caratteri
```

Battendo quindi il tasto RETURN la stringa sarà visualizzata sulla riga successiva. Le parti ombreggiate sono quelle che battete voi, le altre sono la risposta del calcolatore.

```
PRINT "MONDAY"
MONDAY

READY.

PRINT "MAY 12, 1980"
MAY 12,1980

READY.

PRINT "12345"
12345

READY.
```

Provate ora a dare dei comandi di PRINT. Se dopo aver battuto RETURN il calcolatore vi risponde ?SYNTAX ERROR avete probabilmente dimenticato di scrivere PRINT. Se invece ottenere zero avete dimenticato di mettere la stringa tra virgolette. Ripetete la prova sino ad ottenere delle risposte corrette.

Una stringa può essere stampata anche indirettamente assegnado prima il suo valore ad una "variabile di stringa" e poi dando il comando di stampare tale variabile. Una variabile di stringa è un identificatore, o un nome, che può rappresentare una qualunque stringa di caratteri. Tale variabile, per essere riconosciuta dal BASIC, deve essere costituita da una lettera dell'alfabeto, eventualmente seguita da un'altra lettera o da una cifra numerica e poi sempre dal carattere dollaro $. I seguenti sono alcuni esempi di variabile di stringa:

```
A$
MK$
B5$
```

Per stampare una stringa indirettamente è necessario dare due istruzioni. La prima assegna la stringa alla variabile:

```
      V$ = "stringa"
dove: V$ è il nome della variabile di stringa
      stringa è il testo di caratteri
```

La seconda è invece il comando PRINT per stampare il contenuto della variabile:

```
PRINT V$
```

dove:

V$ è il nome della variabile di stringa precedentemente definita.

Ecco alcuni esempi:

```
A$="TAKE THE PROGRAM AND RUN!"

READY.
PRINT A$
TAKE THE PROGRAM AND RUN!

READY.
DY$="TUESDAY"

READY.
PRINT DY$
TUESDAY

READY.
B1$="♠ ♥ ♦ ♠"

READY.
PRINT B1$
♠ ♥ ♦ ♠

READY.
```

Provate ora a stampare il vostro nome in modo indiretto usando una variabile. Battete prima il nome di una variabile con il segno di dollaro, poi il segno eguale e infine, come stringa, il vostro nome tra virgolette. Battete poi il tasto RETURN:

```
NM$="JIMMY OLSON ←— Premere RETURN

READY.
```

Se ottenete la risposta ?TYPE MISMATCH ERROR forse non avete posto il vostro nome tra virgolette e non avete scritto correttamente il nome della variabile (per esempio senza il segno di dollaro $). Battete quindi PRINT seguito dal nome della variabile e premete RETURN. Il vostro nome apparirà sulla riga successiva:

```
NM$="JIMMY OLSON ←— Premere RETURN

READY.
PRINT NM$ ←———— Premere RETURN
JIMMY OLSON

READY.
```

Se apparirà qualcosa di diverso dal vostro nome vi è sicuramente un errore nell'istruzione. Se appare ?SYNTAX ERROR avete probabilmente dimenticato o scritto male PRINT. Se avete dimenticato il segno $ apparirà uno zero, se avete usato un altro nome di variabile di stringa non apparirà niente perchè la seconda variabile non ha contenuto. In ogni caso ripetete ancora la prova.

CALCOLI ARITMETICI

Nel capitolo 4 analizzeremo dettagliatamente tutte le possibilità di programmazione aritmetica, ora vediamo alcuni semplici esempi di addizione, sottrazione, moltiplicazione e divisione. È possibile far eseguire subito una espressione aritmetica e poi stamparne il risultato.

Subito dopo il comando PRINT (o ?) si scrive l'espressione aritmetica da calcolare senza il segno di eguale:

$$\left\{\begin{matrix}\text{PRINT}\\ ?\end{matrix}\right\} \text{ espressione}$$

Ecco alcuni esempi:

```
PRINT 2 + 2
PRINT 5/10
? 2.5
? (100/20) - 16.334
```

Premendo il tasto RETURN il calcolatore eseguirà il calcolo e visualizzerà il risultato.

Provate un esempio: battete PRINT o ? e un semplice calcolo 2 + 2, quindi premete RETURN. Il calcolatore vi risponderà alla riga successiva con il risultato 4:

```
PRINT 2+2 <———————— Premere Return
 4

READY.
```

Se provate ad eseguire gli esempi sopra descritti otterrete sul display le seguenti risposte (ricordatevi che le parti ombreggiate sono quelle battute da voi):

```
PRINT 2+2
 4

READY.
PRINT 5/10
 .5

READY.
? 2.5
 2.5

READY.
? (100/20)+16.334
 21.334

READY.
```

Calcoli aritmetici mediante variabili

Se una variabile è seguita dal segno di dollaro $ ad essa viene associato il valore di una stringa, se invece è priva di tale segno ad essa viene associato un valore numerico.

Anche le variabili numeriche sono definite da una lettera a cui può seguire un'altra lettere o una cifra decimale.

Anche in questo caso potete usare due istruzioni per stampare un valore numerico. Prima dovete associarlo ad una variabile numerica e poi dare l'ordine di stampare la variabile:

```
        V = n
dove:
        V           è una variabile numerica
        n           è un numero
```

e quindi dare l'ordine di stampa:

```
        PRINT V
dove:
        V           è la variabile definita precedentemente
```

Ecco alcuni esempi di assegnazione:

```
A = 1
NM = 2.56
B1 = 1000/10
```

Notate che i valori numerici non sono racchiusi tra virgolette!

Sullo schermo istruzioni e risposte appaiono così:

```
C=100

PRINT C
 100

READY.
F1=1234.78

PRINT F1
 1234.78

READY.
```

MOVIMENTI DEL CURSORE

Con i comandi di movimento del cursore, in modo immediato, è possibile sia muovere il cursore che modificare istantaneamente il testo sullo schermo. **Il cursore si muove su tutto lo schermo nella direzione indicata dalle frecce poste sui tasti.**

Quando si sovrappone ad un precedente carattere non lo modifica, salvo che non si batta poi un nuovo carattere. I comandi del cursore comprendono: CURSOR HOME, CURSOR UP/DOWN e CURSOR LEFT/RIGHT.

Il cursore viene mosso anche da altri tasti: CLEAR SCREEN e INSERT/DELETE. CLEAR SCREEN pulisce lo schermo e pone il cursore nella prima posizione in alto a sinistra detta anche posizione "home" (cioè di riposo). INSERT/DELETE inserisce spazi alla destra del cursore o cancella caratteri alla sinistra.

Se un comando di cursore viene dato in una stringa e cioè dopo l'apertura di virgolette (o dopo un numero dispari di virgolette) esso farà parte della stringa e verrà evidenziato con uno speciale carattere come riportiamo nella Tabella 2-1.

Tabella 2-1: Rappresentazione dei tasti di cursore nelle stringhe

Funzione		Tasto	Simbolo nella stringa
DELETE		INST DEL	(Reverse shift T)
INSERT	Shift	INST DEL	
Home Cursor		CLR SCREEN HOME	(Reverse S)
Clear Screen	Shift	CLR SCREEN HOME	(Reverse Shift S)
Cursor Down		CURSOR	(Reverse Q)
Cursor Up	Shift	CURSOR	(Reverse Shift Q)
Cursor Right		CURSOR	(Reverse])
Cursor Left	Shift	CURSOR	(Reverse Shift])

Riportiamo alcuni esempi di movimenti di cursore contenuti in stringhe:

```
PRINT "1<CRSR→>2<CRSR→>3<CRSR→>4<CRSR→>5"
1 2 3 4 5

READY.
■


PRINT "♠<CRSR←><CRSR→>♥<CRSR→><CRSR→>♦<CRSR←><CRSR→>♠"
♠
♥
♦
♠

READY.
■

PRINT "✱<CLR SCREEN><CRSR↓><CRSR←><CRSR↓><CRSR↓>✱<CRSR↓>✱<CRSR↓>✱<CRSR↓>✱
          <CRSR↓>✱<CRSR↓><CRSR↓><CRSR↓><CRSR↓><CRSR↓>"
    ✱
   ✱
  ✱
 ✱
✱

READY.
■
```

MODO DIFFERITO

Quando si opera in modo differito, le istruzioni sono dapprima caricate nella memoria del calcolatore e solo successivamente saranno eseguite. Per questo motivo ogni istruzione deve iniziare con un numero progressivo.

CARICAMENTO DEL PROGRAMMA

Un programma è un insieme di istruzioni BASIC che guidano il calcolatore alla risoluzione di un problema.

Un programma può essere caricato attraverso la tastiera come anche da una periferica esterna (cassetta o dischi) direttamente in memoria.

Quando caricate un programma dalla tastiera ad ogni istruzione dovete dare un numero progressivo di riferimento. Al termine di ogni linea di istruzioni battete RETURN; essa viene subito memorizzata, ma continua a rimanere presente sullo schermo. Le istruzioni rimangono in memoria sin quando non vengono cancellate o il calcolatore non sia spento. Per evitare quindi di perdere i programmi spegnendo il calcolatore, è necessario avere una memoria esterna come una unità a cassette o a dischi su cui depositare i programmi che si desidera conservare. Per riutilizzare questi programmi basterà richiamarli dalla periferica.

Nel capitolo 4 vedremo come scrivere un programma in BASIC, mentre a questo punto è sufficiente solo sapere come caricarlo, come rileggerlo dalla memoria, come metterlo in esecuzione, come fermare e riprendere la sua esecuzione e come cancellarlo dalla memoria. Come esempio provate a caricare questo piccolo programma che chiamiamo PROVA.

```
10 FOR I=1 TO 800
20 PRINT "A";
30 NEXT I
40 PRINT "UFFA!"
50 END
```

Non preoccupatevi se non riuscite a capirne il significato, cercate invece di capire come il calcolatore opera su un qualunque programma.

Se battendo una linea del programma vi accorgete di avere fatto un errore, oppure è il calcolatore che vi avvisa con ?SYNTAX ERROR, per correggere questa linea basta riscriverla a iniziare dal suo numero e poi battere RETURN alla fine: **la nuova linea sostituisce integralmente la vecchia.**

Vediamo ora come si può "editare" un programma contenuto nella memoria centrale. **Il termine editing ha il significato di redigere e stampare un testo ed è molto usato nella terminologia dei calcolatori.**

Comando LIST

Il comando LIST effettua la visualizzazione del programma, in quel momento presente in memoria, al fine di poterlo leggere, correggere ed eventualmente modificare. Il comando LIST può essere seguito da uno o due numeri di linea:

LIST	(spazio)	lista tutto il programma
	linea	lista una linea
	$linea_1$ — $linea_2$	lista dalla $linea_1$ alla $linea_2$
	— linea	lista dall'inizio alla linea
	linea —	lista dalla linea alla fine

dove: linea — è il numero di linea o riga del programma. $Linea_1$ è minore di $linea_2$. Se un numero di linea non esiste nel programma allora verrà stampata una linea bianca.

Se il programma è più lungo di quanto può stare sullo schermo, il comando LIST farà scorrere verso l'alto ("scroll") le righe. È preferibile quindi visualizzare sullo schermo solo quelle righe di programma che interessano usando i parametri di linea dopo il comando LIST.

Ecco infatti alcuni esempi:

LIST	lista l'intero programma
LIST 50	lista solo la riga 50
LIST 60 — 100	lista tutte le righe dalla 60 alla 100 incluse
LIST — 140	lista tutte le righe dall'inizio alla 140 inclusa
LIST 20000 —	lista dalla 20000 alla fine

Provate a listare il programma PROVA: battete LIST e poi il tasto RETURN.

```
LIST

 10 FOR I=1 TO 800
 20 PRINT "A";
 30 NEXT I
 40 PRINT UFFA!
 50 END
READY.
```

Noterete che il cursore scompare durante la stampa del programma per riapparire solo dopo l'avviso di READY e avvisarvi che il calcolatore è ritornato in modo immediato.

Esercitatevi ora a listare solo una parte del programma per esempio solo la riga 10:

```
LIST 10

 10 FOR I=1 TO 800

READY.
```

Oppure dalla riga 20 alla 40:

```
LIST 20-40

 20 PRINT "A";
 30 NEXT I
 40 PRINT UFFA!
READY.
```

Continuate da soli a provare tutte le possibilità del comando LIST.

Comando di esecuzione RUN

Il comando RUN pone in esecuzione il programma in quel momento presente in memoria. Battete RUN e il tasto RETURN. Il formato previsto per il comando RUN è il seguente:

RUN	{ spazio	pone in esecuzione dall'inizio del programma
	{ linea	pone in esecuzione dalla riga indicata

Provate ad eseguire il programma PROVA battendo RUN e poi RETURN. Un attimo dopo lo schermo si riempirà di 800 lettere A e terminerà con un UFFA! Notate che anche in questo caso, durante l'esecuzione del programma, il cursore scompare per riapparire solo alla fine quando il calcolatore ritorna in modo immediato.

```
RUN
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
"UFFA!"

READY.
```

Potete indicare al calcolatore di iniziare l'esecuzione di un programma da una qualunque riga indicandone il numero dopo il comando RUN. Per esempio a partire dalla riga 40:

```
RUN 40
UFFA!

READY.
```

Attenzione però che non sempre un programma può essere posto in esecuzione da una riga qualunque. L'esempio PROVA non può partire infatti dalle righe 20 e 30 e in tal caso avreste subito un avviso di errore.

Come fermare l'esecuzione

Un programma può essere fermato durante la sua esecuzione e questo prende il nome di "break". Il comando relativo si dà battendo il tasto STOP. Il calcolatore vi avviserà subito indicandovi a quele riga avete fermato il programma:

```
RUN
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAA ◄──────────────────────────── Premere STOP
BREAK IN 40 ◄──────────────────────────── Messaggio di "BREAK"
READY.
```

Durante un break di programma il calcolatore ritorna in modo immediato così che potete dare comandi supplettivi come LIST, LOAD, SAVE e VERIFY oppure qualunque altra istruzione compatibile con il modo immediato come, per esempio, cambiare il valore a delle variabili prima di riprendere l'esecuzione del programma.

Continuazione di un programma interrotto

L'istruzione "continue" CONT effettua la ripresa dell'esecuzione di un programma precedentemente fermato. Se battiamo CONT per riprendere il programma PROVA otteniamo:

```
RUN
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAA ◄──────────────────────────── Premere STOP
BREAK IN 40
READY.
CONT ◄──────────────────────────────────── Comando di CONT
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Attenzione però che il comando CONT non fa riprendere l'esecuzione di un programma fermato a causa di un errore!

Come cancellare un programma dalla memoria

Un programma in memoria può essere cancellato mediante l'istruzione NEW subito seguita da RETURN. Questa procedura non cancella però ciò che è sullo schermo. Fate quindi attenzione ad usare il comando NEW perchè potreste accidentalmente perdere un programma senza accorgervene. Provate a cancellare il programmino PROVA:

```
LIST ←————————— LIST del programma

 10 FOR I=1 TO 800
 20 PRINT "A";
 30 NEXT I
 40 PRINT "UFFA!"
 50 END
READY.
NEW ←————————— Cancellazione del programma

READY.
LIST ←————————— Nuovo LIST
     ←————————— Nessun programma in memoria
READY.
```

È buona norma, prima di caricare un nuovo programma, dare il comando NEW per cancellare un eventuale vecchio programma ed evitare quindi spiacevoli sovrapposizioni.

INSIEMI DI CARATTERI STANDARD E ALTERNATIVO

Ogni calcolatore CBM ha un insieme di caratteri "standard" ed uno "alternativo". Questi insiemi di caratteri sono riportati nell'Appendice A.

Al momento dell'accensione i calcolatori PET operano subito con l'insieme standard; i calcolatori CBM operano invece con l'insieme alternativo. Per cambiare l'insieme di caratteri dovete forzare un numero con il comando POKE nella posizione di memoria 59468. Se questo numero è 12 l'insieme selezionato e quello standard, se invece è 14 l'insieme selezionato è quello alternativo.

POKE 59468,12	attiva l'insieme di caratteri standard
POKE 59468,14	attiva l'insieme di caratteri alternativo

Programmazione con l'insieme di caratteri alternativo

Quando usate l'insieme alternativo e state scrivendo un programma in BASIC, in un calcolatore CBM, non potete selezionare le lettere maiuscole con il tasto SHIFT abbassato.

I calcolatori CBM assumono che sia sempre presente l'insieme di caratteri standard per cui, se premete SHIFT, l'interprete BASIC riconosce i caratteri standard shiftati che non sono le lettere maiuscole.

Facciamo un esempio. In alcuni casi potreste volere battere le lettere maiuscole abbassando SHIFT, come per la F di questa istruzione:

```
10 For I=1 To 10 Step 2
```

Il calcolatore CBM assume però che i caratteri appartengono all'insieme standard, per cui se il testo SHIFT è abbassato riconosce altri caratteri:

```
10 —OR ↖=1 TO 10 ♥TEP 2
```

Ovviamente questa istruzione è errata e verrà rifiutata.

Solo nell'ambito delle stringhe non avrete alcun errore con i caratteri alternativi shiftati.

IMPIEGO DELLE UNITA' A CASSETTE

Le unità periferiche a cassette o a dischetti permettono di memorizzare sia programmi che dati. I calcolatori CBM, con dimensioni piccole di memoria, normalmente si interfacciano a cassette mentre quelli con memorie più grandi si collegano a dischetti. Diamo ora le istruzioni per operare con le unità a cassette.

Per prima cosa assicuratevi che una cassetta sia caricata correttamente nel drive # 1 come abbiamo già visto nel capitolo 1. Riavvolgete il nastro al suo inizio mediante il tasto REWIND e poi premete STOP. Controllate che tutti gli altri tasti siano nella posizione spento (cioè alzati).

Come primo esempio caricheremo su cassetta il programma PROVA. Le istruzioni ombreggiate sono le vostre mentre le altre sono quelle di risposta del calcolatore.

Comando SAVE di un programma

Se memorizzate i vostri programmi su una cassetta siete sicuri di non perderli quando spegnerete il calcolatore. Ricordate però che il comando SAVE salva solo il programma in quel momento presente in memoria.

Il formato del comando SAVE è il seguente:

SAVE "nome del programma"	il programma è salvato sulla cassetta nel drive # 1
SAVE "nome del programma", 1	il programma è salvato sulla cassetta nel drive # 1
SAVE "nome del programma", 2	il programma è salvato sulla cassetta nel drive # 2

Il numero # 1 o # 2 indica su quele drive si vuole salvare il programma. L'indicazione # 1 può essere omessa perchè il calcolatore considera il primo drive

come privilegiato o di obbligo (in inglese "default"), ma se volete andare sul secondo dovete indicarlo esplicitamente.

Per far pratica caricate quindi il programma PROVA su una cassetta (drive # 1). Battete prima il testo alla tastiera, poi date il comando SAVE "PROVA" e premete RETURN. Il calcolatore risponde di predisporre il registratore PRESS PLAY & RECORD ON TAPE #x.

```
SAVE "PROVA"

PRESS PLAY & RECORD ON TAPE #1
```

Premete allora contemporaneamente PLAY e RECORD. Il registratore si mette in movimento. Se per errore avete premuto solo PLAY il registratore non si pone in registrazione. Sia voi che il calcolatore crederete di registrare, ma non sarà assolutamente così.

All'inizio della registrazione il calcolatore vi avvisa con OK e WRITING PROVA; alla fine vi da READY e riappare il cursore.

```
SAVE "PROVA"

PRESS PLAY & RECORD ON TAPE #1
OK <------------------------------ Testo PLAY premuto sul driver # 1

WRITING PROVA
READY.
```

Attenzione: se uno dei quattro tasti REC, REW, FFWD o PLAY è premuto durante l'esecuzione di una istruzione SAVE il calcolatore inizierà subito a "scrivere". Premete quindi, sempre per prima cosa, il tasto STOP e assicuratevi di ricevere il messaggio PRESS PLAY & RECORD ON TAPE # x. **Attenzione ancora a premere correttamente PLAY e RECORD.**

Dopo una operazione SAVE controllate che il programma sia stato correttamente registrato con una istruzione di verifica VERIFY.

Comando VERIFY di un programma

Questo comando verifica l'esattezza di una registrazione. Esso simula un comando di caricamento LOAD dalla periferica e controlla l'eguaglianza tra il programma registrato e il programma ancora contenuto nella memoria centrale. Se trova una diversità avvisa subito con un messaggio. Vi consigliamo di verificare sempre un programma subito dopo una operazione SAVE.

Il formato di VERIFY è:

VERIFY	verifica il primo programma che incontra sulla cassetta # 1
VERIFY "nome del programma", 1	verifica il programma nominato dalla cassetta # 1
VERIFY "nome del programma", 2	verifica il programma nominato dalla cassetta # 2

Provate allora a verificare il programma PROVA che avete appena registrato:

1. Riavvolgete il nastro all'inizio o al punto da cui inizia la vostra registrazione.
2. Premete il tasto STOP.
3. Battete il comando VERIFY. In questo caso non è necessario che diate ne il nome del programma ne il numero di drive. Il numero del drive lo dovete dare solo se è il secondo.

La successione di comandi e messaggi di riposta sul display sarà allora la seguente:

```
SAVE "PROVA"

PRESS PLAY & RECORD ON TAPE #1
OK

WRITING PROVA
READY.  <------------------------------  Riavvolgere il nastro e posizionarlo
VERIFY PROVA                             prima dell'inizio del programma PROVA

PRESS PLAY ON TAPE #1
OK  <------------------------------  Premere PLAY sulla cassetta # 1

SEARCHING FOR PROVA
FOUND PROVA
VERIFYNG
OK

READY.
```

Quando appare il messaggio PRESS PLAY ON TAPE # 1 premete il tasto PLAY sul drive # 1. L'unità a cassette ricercherà il programma PROVA e vi avviserà con i messaggi OK e SEARCHING FOR PROVA e una volta trovato vi dirà FOUND PROVA. Durante la verifica sarete avvisati con VERIFYING e se la registrazione è corretta avrete un OK finale. Diversamente in caso di errori avrete il messaggio ?VERIFY ERROR. In quest'ultimo caso riavvolgete il nastro e riprovate la lettura con VERIFY. Se la segnalazione di errore persiste allora sicuramente il programma PROVA è stato mal registrato. Registratelo nuovamente con il comando SAVE, verificatelo con VERIFY e se alla fine riscontrate ancora errori è molto probabile che il nastro della cassetta in quel punto sia difettoso. Provate a registrare su una diversa porzione di nastro o al limite cambiate cassetta.

Comando LOAD di un programma

Il comando LOAD carica un programma nella memoria centrale del calcolatore. Per caricare un programma dal drive # 1 potete usare uno dei due formati seguenti:

```
LOAD "nome del programma"
LOAD "nome del programma", 1
```

Quando si dà il comando LOAD non è sempre necessario specificare il nome del programma. Se lo si specifica il calcolatore caricherà soltanto quel programma. Diversamente caricherà il primo programma che incontra sul nastro. Analogamen-

te non è richiesto di specificare il numero # 1 di drive, mentre è obbligatorio indicare # 2 se si lavora con la seconda unità a cassette. Per il calcolatore PET 2001 la cassetta # 1 è quella assemblata al suo interno mentre per gli altri calcolatori è quella collegata all'interfaccia posteriore. Ripetiamo ancora che se si lavora con un drive solo o con il primo non è necessario specificare il numero # 1, mentre bisogna farlo se si lavora con l'eventuale secondo drive.

Sono eguali infatti:

```
    LOAD "PROVA"
o   LOAD "PROVA", 1
```

Facciamo un esempio di come si può caricare il nostro programma PROVA:

```
LOAD "PROVA"

PRESS PLAY ON TAPE #1 <--- Premere PLAY sulla cassetta # 1
OK

SEARCHING FOR PROVA
FOUND PROVA
LOADING
READING
▒
```

A seguito del comando LOAD il calcolatore risponde PRESS PLAY ON TAPE # 1. Se tutto funziona correttamente il calcolatore vi avvisa con OK e poi con SEARCHING FOR PROVA. Per ogni programma che viene incontrato sul nastro appare l'indicazione FOUND seguita dal nome, ma solo quando sarà individuato PROVA avverrà il caricamento LOADING e alla fine apparirà READY.

Al termine di una operazione di caricamento si può lasciare abbassato il tasto PLAY per un eventuale altro caricamento. Ma attenzione se dovete invece fare un'operazione SAVE commettereste un grave errore perchè in tal caso dovete premere anche RECORD. **Il calcolatore "sente" che un tasto è premuto, ma non sa riconoscere quale.**

Alcune volte si ha difficoltà a caricare un programma nella memoria. Può accadere che il calcolatore riconosca il nome del programma, ma non lo carichi affatto. Provate a ripetere anche più volte l'operazione di caricamento. Se proprio non riuscite può significare o che la precedente operazione SAVE è stata mal effettuata o che l'unità a cassette è guasta.

Prima di dare un comando LOAD ricordatevi di riavvolgere il nastro sino ad un punto poco prima di quello da cui inizia la registrazione del programma. In caso diverso il calcolatore cercherà inutilmente di individuare il vostro programma fino alla fine del nastro. Desideriamo darvi un altro consiglio: se il vostro programma è registrato verso la fine del nastro e voi iniziate la sua ricerca proprio dall'inizio, passerà molto tempo prima che il calcolatore legga tutto il nastro e trovi il programma. Vi consigliamo allora di usare la finestrella, che è posta al centro di ogni cassetta, per avere una indicazione seppure approssimata della posizione del

nastro. Così con i comandi REW e FFWD vi potete portare un pò prima del punto presumibile di inizio del vostro programma. Da questo punto approssimativo lasciate poi che sia il comando LOAD a trovare il programma.

La finestrella di cui si è accennato sopra è ripartita in 100 tacche, segnate di 10 in 10, e attraverso essa si vede il nastro avvolto. A seconda dello spessore dell'avvolgimento si ha una diversa indicazione della posizione del nastro.

Una volta che il programma è caricato in memoria è buona norma "listarlo" (cioè leggerlo) con il comando LIST e poi, se ogni cosa ci sembra corretta, porlo in esecuzione con il comando RUN.

Comando LOAD & RUN

Il comando LOAD & RUN carica ed esegue automaticamente il primo programma incontrato sulla cassetta # 1. Tale comando viene dato premendo il tasto RUN/STOP shiftato.

LOAD & RUN esiste solo con le versioni BASIC < 3.0 (versioni 1.x, 2.x e 3.x).

Questo comando opera solo con il drive # 1 e non permette di indicare il nome del programma. Di conseguenza carica ed esegue solo il primo programma che incontra. Per quanto concerne i messaggi di avviso esso visualizza gli stessi messaggi di LOAD:

```
LΓ                                          ← Premre il taso RUN/STOP shiftato

PRESS PLAY ON TAPE #1                       ← Premere PLAY sulla cassetta # 1
OK

FOUND PROVA
LOADING
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  ← Esecuzione automatica del programma
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
......
```

Il comando LOAD & RUN, che viene dato premendo RUN/STOP shiftato, può essere dato solo in modo immediato. Se lo date durante l'esecuzione di un programma esso ne interrompe l'esecuzione perchè funziona come STOP.

IMPIEGO DELLE UNITA' A DISCO

Spiegheremo ora come memorizzare un programma su un dischetto e poi come richiamarlo e porlo in esecuzione.

Per svolgere queste funzioni non è necessario conoscere la programmazione BASIC ne i comandi specifici di una unità a disco come vedremo invece nel capitolo 6. Le prime operazioni che si devono imparare sono: come "inizializzare" i drive, come stampare (listare) il contenuto della "directory" di un dischetto e come memorizzare, caricare ed eseguire un programma da dischetto.

DOS VERSIONE 1.x E 2.x

DOS significa "Disk Operating System" ed è l'insieme di programmi che sovrintendono alla gestione delle periferiche a disco.

La Commodore ha presentato versioni successive di DOS che indichiamo con una numerazione decimale. La prima versione DOS 1.x (x è un numero tra 1 e 9 che rappresenta una sottoversione) è compatibile con alcuni modelli di drive CBM 2040 ed è usata con il BASIC < 3.0.

La versione DOS più sviluppata è invece la 2.x. Essa presenta comandi di gestione del disco notevolmente semplificati e migliorie nella memorizzazione dei dati. Sono stati anche corretti alcuni errori che si erano riscontrati nella versione 1.x. Il DOS 2.x può operare con qualunque versione di BASIC, mentre il DOS 1.x opera solo con il BASIC < 3.0. Il DOS 1.x potrebbe anche operare con il BASIC 4.0, ma alcuni comandi al disco del BASIC 4.0 possono causare degli errori di sintassi (disk syntax error).

Tutte le unità a disco CBM 8050 impiegano il DOS versione 2.x con il BASIC 4.0. Invece i modelli CBM 2040 usano o il DOS 1.x con il BASIC < 3.0 o il DOS 2.x con il BASIC 4.0.

COME CARICARE UN PROGRAMMA DA DISCHETTO CON IL BASIC < 3.0.

La procedura da seguire in questo caso è la seguente:

1. **Aprire un "file" (archivio) logico sulla periferica.**
2. **Inizializzare il (o i) drive.**
3. **Listare (stampare) la "directory". Questo punto è opzionale, ma permette l'esatta verifica di tutti i nomi di programma.**
4. **Caricare il programma.**

Per dimostrare quanto detto impiegheremo il dischetto TEST/DEMO fornito assieme all'unità a disco dalla Commodore. Iniziate quindi a inserire il dischetto TEST/DEMO nel drive 0 come abbiamo già visto nel capitolo 1.

Apertura di un file logico e di una periferica

Prima di caricare un programma da un dischetto è necessario "aprire", cioè stabilire, una linea di comunicazione logica tra il calcolatore e l'unità a disco. Il comando OPEN, da inserirsi in modo immediato, ha appunto questo scopo.

Il formato di un comando OPEN è il seguente:

```
OPEN 1, 8, 15
```

Come inizializzare un drive di unità a disco (BASIC < 3.0)

Quando si usa il BASIC < 3.0 ogni drive deve essere inizializzato ogni volta che si inserisce in esso un dischetto. Questo deve essere fatto anche quando si sposta da un drive all'altro uno stesso dischetto. Ricordiamo che un drive deve contenere un dischetto prima dell'operazione di inizializzazione.

Il formato del comando INITIALIZE (o la sua forma abbreviata "I") è il seguente:

```
PRINT # file, "{INITIALIZE}dr"
              {    I     }
```

dove:

file è il numero logico di file usato nel comando OPEN
dr è il numero di drive 0 o 1

Per inizializzare il drive 0 contenente il dischetto TEST/DEMO, battete:

```
PRINT # 1, "I0"
```

In corrispondenza dovete vedere l'indicatore luminoso del drive 0 accendersi per il breve tempo che il motorino gira. Quando la spia si spegne il drive è inizializzato.

Per inizializzare il drive 1 inserite invece in esso il dischetto e battete:

```
PRINT # 1, "I1"
```

Per inizializzare ambedue i drive, battete:

```
PRINT # 1, "I"
```

Caricamento della directory di un dischetto (DOS 1.x)

Se desiderate conoscere quali programmi sono registrati su un dischetto dovete leggere la sua "directory" che rappresenta appunto l'indice di quanto contiene. Se conoscete già la directory passate allora al paragrafo successivo.

Per leggere una directory dal drive 0, nell'ambito del BASIC < 3.0, battete:

```
LOAD "$0", 8
```

Se il dischetto è nel drive 1, battete invece:

```
LOAD "$1", 8
```

Se non specificate il numero di drive verranno lette ambedue le directory:

```
LOAD "$", 8
```

A conferma del caricamento della directory appariranno sullo schermo i seguenti avvisi:

```
LOAD"$0",8

SEARCHING FOR $0
LOADING
READY.
```

Dopo l'avviso READY battete LIST e premete RETURN.
Sullo schermo appariranno in successione i nomi dei vari programmi. Se sono più di 25 i primi nomi scorreranno verso l'alto fuori dello schermo per far posto alle righe successive. Per rallentare lo scorrimento delle righe ("scroll") tenete abbassato il tasto REVERSE o il tasto ← nei calcolatori commerciali. Per fermare la visualizzazione premete il tasto STOP.

Potete rivedere la vostra directory dando nuovamente il comando LIST.

Ecco un esempio di directory del dischetto TEST/DEMO per il modello CBM 2040:

```
LIST

0 [illegible]
6     "DOS SUPPORT 4.0"   PRG
27    "DUM 3.4"           PRG
1     "  DISK DATA  "     SEQ
15    "DIAGNOSTIC BOOT"   PRG
10    "COPY DISK FILES"   PRG
4     "CHECK DISK"        PRG
10    "PET DISK"          PRG
10    "DISK DISPLAY"      PRG
3     "DISK COMM"         PRG
2     "DISK COMM2"        PRG
3     "DISK COMM3"        PRG
4     "DISK WRITE"        PRG
4     "DISK READ"         PRG
2     "DISK OVERLAYS"     PRG
5     "DISK DIR"          PRG
7     "  PET DATA  "      SEQ
34    "RANDOM 1.00"       PRG
27    "PRINTER DEMO"      PRG
12    "SEQUENTIAL 1.00"   PRG
484 BLOCKS FREE.
```

La prima riga è il titolo o intestazione. Le altre riportano il nome del programma e l'indicazione di quanti "blocchi" occupa ogni singolo programma. Le virgolette che racchiudono i nomi non fanno parte del nome stesso. I blocchi sono un'unità di misura dei dischi e rappresentano le parti in cui esso viene diviso. Al termine della directory viene data una informazione molto utile che ci indica quanti blocchi sono ancora disponibili per altre registrazioni. Infine sulla destra viene specificato se a ogni nome corrisponde un programma (PRG) perchè potremmo anche aver registrato archivi o file di dati. Non preoccupatevi troppo di tutta l'informazione che una directory vi dà: per ora cercate solo di vedere se ritrovate i vostri programmi.

Il comando LOAD (BASIC < 3.0)

Per leggere un programma da dischetto e caricarlo in memoria si dà il comando LOAD con il seguente formato:

```
LOAD "0: nome del programma", 8
```

Dove 0 è il numero del drive che è separato dal nome del programma da due punti (:) e ambedue sono posti tra virgolette. Se non viene specificato il numero di drive il sistema cercherà il programma su ambedue i drive a condizione, ovviamente, che siano stati inizializzati.

Per caricare il primo programma presente sul dischetto TEST/DEMO, inserito nel drive 0, battete:

```
LOAD "0:DOS SUPPORT 4.0",8
```

Quando premete RETURN vi appare sullo schermo il seguente messaggio, mentre l'indicatore luminoso LED del drive si accende e l'unità produce un leggero ronzio:

```
LOAD "0:DOS SUPPORT 4.0",8

SEARCHING FOR 0 DOS SUPPORT 4.0
LOADING
READY.
```

Dopo il messaggio SEARCHING FOR appare sempre il numero del drive e il nome del programma. Quando poi il programma è caricato si spegne l'indicatore LED e cessa il ronzio mentre sullo schermo appaiono il segnale READY e il cursore. Il programma può quindi essere listato e posto in esecuzione.

Il comando LIST (BASIC < 3.0)

Un programma caricato in memoria può essere listato mediante un comando LIST. Se si desidera listarne solo una parte vi ricordiamo che è sufficiente aggiungere a LIST le indicazioni dei numeri di linea.

Il comando RUN (BASIC < 3.0)

Anche questo è un comando che già conosciamo. Battete RUN seguito da RETURN.

Come preparare un dischetto nuovo nel BASIC < 3.0

Un dischetto nuovo o vergine (in inglese "blank") deve essere preparato o "formattato" prima di poter essere impiegato. Descriviamo allora la procedura per dare formato a un dischetto nuovo o ridarlo ad uno vecchio che intendiamo riusare.

"Aprite" il drive come segue:

```
OPEN 1, 8, 15
```

Per dare formato ad un dischetto e inizializzare il drive si usa il comando NEW nel modo seguente:

PRINT # file, " { N / NEW } dr: nome del disco, id"	
dove: file	è il numero logico del file usato nel comando OPEN
dr	è il numero di drive 0 o 1
nome del disco	è ovviamente il nome dato a quel particolare dischetto
id	è un identificatore di due caratteri

Nel nostro esempio il dischetto è inserito nel drive 0, il nome è YAK e il numero di identificazione è 01. Battete allora i comandi OPEN e NEW:

```
OPEN 1,8,15
PRINT#1,"N0:YAK,01"
```

Battete RETURN. Come al solito l'indicatore LED si accende e alla fine sullo schermo appare READY accompagnato dal cursore. Potete quindi svolgere altre funzioni come caricare la directory:

```
LOAD "$0",8
```

Abbiamo indicato $0 perchè il dischetto è nel drive 0; diversamente sarebbe stato $1. Quando la directory è caricata possiamo visualizzarla, battendo:

```
LIST
```

Ogni directory inizia con il suo titolo e termina con l'indicazione dei blocchi liberi che in questo caso sono ovviamente tutti:

```
LIST

 0 [illegible]
 670 BLOCKS FREE.
READY.
%
```

A questo punto il dischetto è formattato e potete quindi registrare su esso i vostri programmi o file.

Il comando SAVE (BASIC < 3.0)

Il comando SAVE per registrare un programma su un dischetto è molto simile a quello per registrare su una cassetta. Le differenze risiedono nella sintassi, cioè nel

formato, che nel nostro caso è il seguente:

```
SAVE "dr: nome", 8
```

dove:

dr è il numero del drive 0 o 1
nome è il nome del programma

Il numero del drive (0 o 1) non può essere omesso. Il numero 8 è importante perchè avvisa il sistema che la registrazione deve andare su disco. Se il numero 8 viene omesso il sistema cercherà di registrare sulla cassetta 1.

Provate ora a registrare il vostro programma PROVA su un dischetto nel drive 0. Inserite un dischetto già con formato nel drive 0; quindi date OPEN e INITIALIZE al drive, se non lo avete già fatto in precedenza. Caricate alla tastiera il programma PROVA e date il comando SAVE e RETURN.

```
OPEN 1,8,15 <------------ Apertura (OPEN) del driver
PRINT#1,"I0" <----------- Inizializzazione del drive 0

10 FOR I=1 TO 800
20 PRINT "A"
30 NEXT I                 Registrazione del programma PROVA
40 PRINT "UFFA!"
50 END

SAVE "0:PROVA",8 <------- Conferma

READY.
```

Mentre il programma viene "salvato" il LED del drive si accende e sul video mancano READY e il cursore. Al termine della registrazione, e a conferma che tutto è regolare, ritornano appunto sul video READY e il cursore. Come già detto in precedenza, dopo una registrazione, è buona norma verificare se essa è avvenuta correttamente con un comando VERIFY.

Il comando VERIFY (BASIC < 3.0)

Il procedimento per verificare un programma sia da cassetta che da dischetto è analogo. Il calcolatore confronta quello che legge dalla memoria esterna con quello che è ancora nella sua memoria centrale. Se trova delle discordanze avvisa con un messaggio di errore.

Verificate sempre un programma dopo la sua registrazione! Il formato del comando VERIFY è simile a quello di SAVE:

```
VERIFY "dr: nome", 8
```

dove:

dr è il numero del drive 0 o 1
nome è il nome del programma appena registrato

Appena premete RETURN il calcolatore inizia il processo di verifica e vi avvisa con i messaggi SEARCHING FOR PROVA, VERIFYNG e OK. Se invece di OK appare una segnalazione di errore allora provate a riverificare e se l'errore permane allora dovete ripetere il SAVE e poi controllare con VERIFY. La registrazione di un programma, che non supera la verifica, non vi serve a nulla perchè non potrete mai leggere tale programma!

```
SAVE "0:PROVA",8          ← SAVE il programma PROVA
READY
VERIFY "0:PROVA",8        ← Verifica PROVA
SEARCHING FOR 0:PROVA
VERIFYING                 ← Programma verificato
OK

READY.
```

Un formato più sintetico per dare il comando VERIFY permette di usare un asterisco, invece del nome del programma, quando il comando viene dato subito dopo l'operazione SAVE:

```
VERIFY "*", 8
```

COME CARICARE UN PROGRAMMA DA DISCHETTO CON IL BASIC 4.0

A differenza di quanto avviene nel BASIC < 3.0, nell'ambito del BASIC 4.0 non è sempre necessario inizializzare il drive.

Le operazioni necessarie per caricare un programma da un dischetto sono due:

1. Listare la directory.
2. Caricare il programma.

Per provare queste operazioni potete usare un qualunque dischetto contenente dei programmi. Qui di seguito negli esempi noi usiamo il dischetto DEMO.

Il BASIC 4.0 inizializza automaticamente il drive, prima di caricare un programma, mentre carica la directory. Tuttavia usando dischetti che abbiano lo stesso numero di identificazione (cioè il numero posto dopo il nome nella intestazione "header") **dovete effettuare manualmente la inizializzazione, se scambiate uno di questi dischetti con un altro, perchè il calcolatore non può sapere del cambiamento di dischetto.**

Caricamento della directory del dischetto (BASIC 4.0)

Il comando DIRECTORY carica la directory del dischetto e la visualizza sul display.

Battete quindi:

```
DIRECTORY D0   carica e lista la directory dal drive 0
DIRECTORY D1   carica e lista la directory dal drive 1
DIRECTORY      carica e lista le directory da ambedue i drive
```

Se il drive non è specificato il sistema carica e lista le directory da ambedue i dischetti inseriti nei due drive.

Un esempio di richiamo e stampa di directory dal drive 0 il seguente:

```
DIRECTORY D0

 0 [illegible]
 5    "UNIVERSAL WEDGE"  PRG
 8    "UNIT TO UNIT"     PRG
 3    "CHANGE 8050"      PRG
 11   "COPY 2040 - 8050" PRG
 27   "PRINTER DEMO"     PRG
 12   "SEQUENTIAL"       PRG
 11   "PERFORMANCE TEST" PRG
 5    "CHECK DISK"       PRG
 17   "LOGIC DIAGNOSTIC" PRG
 1953 BLOCKS FREE.
READY.
```

Il comando DLOAD (BASIC 4.0)

Il comando per caricare un programma da dischetto con il BASIC 4.0 è il seguente:

DLOAD "nome del programma" carica dal drive 0

DLOAD "nome del programma", D0

DLOAD "nome del programma", D1 carica dal drive 1

Il nome del programma è obbligatorio e deve essere scritto tra virgolette. In mancanza del numero di drive il calcolatore impone il valore 0 e se il programma non è presente appare l'avviso ?FILE NOT FOUND.

Provate queste istruzioni sul vostro calcolatore. Per esempio caricate il dischetto DEMO, o altro dischetto, nel drive 0 e battete:

```
DIRECTORY D0
```

Il cursore sparisce momentaneamente mentre il dischetto è inizializzato e la directory è caricata in memoria.

Caricate quindi il secondo programma della directory (che può anche avere un nome diverso da quello che appare in questo testo):

```
DLOAD "UNIT TO UNIT"
```

Sullo schermo appariranno i messaggi di avviso di ricerca e caricamento del programma:

```
DLOAD "UNIT TO UNIT",D0

SEARCHING FOR UNIT TO UNIT
LOADING
READY.
```

Il comando LIST (BASIC 4.0)

Questo comando è quello che già conoscete ed ha le stesse caratteristiche sia nel BASIC < 3.0 che nel BASIC 4.0 ed in tutte le versioni di DOS.

Il comando RUN (BASIC 4.0)

Per eseguire il programma battete RUN e poi premete RETURN.

```
RUN
```

Il comando LOAD & RUN (BASIC 4.0)

Il comando sintetico LOAD & RUN carica e pone in esecuzione automaticamente il primo programma incontrato sul dischetto nel drive 0. Per dare questo comando basta premere il tasto RUN/STOP contemporaneamente al tasto SHIFT (SHIFT è il tasto che permette di ottenere i caratteri superiori della tastiera).

Attenzione però che il comando LOAD & RUN è previsto solo per le versioni di BASIC 4.0 (versione 4.x).

Per dare il comando LOAD & RUN premete allora il tasto RUN/STOP nella posizione superiore (shift). Sullo schermo appariranno alcuni messaggi prima dell'esecuzione del programma:

```
                    ←———————— Premere il tasto RUN shiftato
DL"*

SEARCHING FOR 0 *
LOADING
                    ←———————— Inizio esecuzione del programma
```

Ricordiamo che il tasto RUN può essere premuto solo se il calcolatore è in modo immediato. Diversamente se premuto durante l'esecuzione di un programma esso opera come un comando di STOP e ferma il programma.

Come preparare un dischetto nuovo nel BASIC 4.0

Anche nel BASIC 4.0 un dischetto nuovo deve essere "formattato" prima di poter essere usato. **La procedura per formattare un dischetto nuovo, o riformattare uno vecchio è la seguente.**

Inserite il dischetto nel drive 0.

Battete il comando di formattazione HEADER:

```
        HEADER "nome del disco", Dx, I22
dove:
        nome del disco  è il nome che viene dato a quel particolare dischetto
        x               è il numero del drive 0 e 1
        22              è un numero di identificazione di due cifre
```

Provate a dare il formato al vostro dischetto con il comando HEADER:

```
HEADER "YAK", D0, I01
```

Abbiamo dato al disco il nome YAK e lo abbiamo inserito nel drive 0. Se lo avessimo inserito nell'altro drive avremmo dovuto indicare D1 invece di D0. Il numero di identificazione è 01, ma avremmo potuto usare qualunque altre due cifre o lettere.

Quando premerete RETURN il calcolatore vi chiede una conferma (per evitare errori di identificazione del disco) ARE YOU SURE?. Se volete confermare battere Y per sì o N per no e poi RETURN. Se avete confermato il calcolatore svolge il suo lavoro e poi visualizza READY; se invece avete annullato il comando HEADER il calcolatore esce dalla procedura di formattazione.

```
HEADER "YAK", D0, I01
ARE YOU SURE ?Y ←———— Premere Y per si
READY.
*
```

A questo punto potete usare il dischetto come supporto di dati e programmi.

Il comando DSAVE (BASIC 4.0)

Questo comando permette la memorizzazione di programmi su dischetto. Esso deve essere dato in modo immediato ed ha le seguenti caratteristiche:

```
     DSAVE "nome del programma", Dx
dove:
     nome del programma        è appunto il nome del programma
     x                         è il numero di drive 0 e 1
```

Il nome del programma è obbligatorio e deve essere racchiuso tra virgolette. Il numero di drive è opzionale e viene inteso come 0 se non specificato.

Provate a memorizzare il programma PROVA, che in precedenza avete caricato dalla tastiera, sul drive 0.
Battete infatti:

```
DSAVE "PROVA",D0
```

Appena premete RETURN il LED del drive si accende e il cursore sparisce dallo schermo. Alla fine ritornano READY e cursore:

```
DSAVE "PROVA",D0

READY.
*
```

Verificate ora il vostro programma.

Il comando VERIFY (BASIC 4.0)

Questo comando è identico a quello della versione BASIC < 3.0 per cui potete seguire le procedure già viste in precedenza.

IMPIEGO DELLE STAMPANTI CBM

Le stampanti CBM svolgono una funzione complementare al display video in quando permettono di stampare programmi e dati su carta. Esse sono controllate in maniera analoga alle unità, a cassette o a dischetti, e ricevono i comandi o in modo immediato o come istruzioni di un programma. Descriviamo ora unicamente i comandi in modo immediato rimandando al capitolo 6 i comandi da programma. **Precisiamo ancora che l'impiego delle stampanti è eguale per ambedue le versioni di BASIC < 3.0 e 4.0.**

Il comando OPEN

Prima di mandare dati a una stampante è necessario stabilire una linea di comunicazione logica tra essa e il calcolatore. Come già visto con altre periferiche questo si ottiene con l'istruzione OPEN:

```
OPEN x,4
```

dove:

x è un intero tra 1 e 255

Ecco alcuni esempi:

```
OPEN 1,4

OPEN 4,4

OPEN 250,4
```

Il comando CMD

Qualora la stampante sia stata "aperta", i testi in uscita le possono essere mandati con il comando:

```
CMD 1
```

Ovviamente il numero 1, o un altro, deve corrispondere al numero x che si è dato nella precedente istruzione OPEN. Se viene erroneamente usato un numero diverso apparirà il messaggio ?FILENOT OPEN ERROR e si dovrà quindi ripetere l'apertura della stampante con OPEN e ribattere CMD.

Ecco alcuni esempi di comandi PRINT:

```
OPEN 1,4:CMD 4
PRINT "TISHNICK"


OPEN 2,4
CMD 2, "MY PET BITES"


OPEN 3,4
PRINT#3, "54321"
```

Uscita di testi su stampante

Quando la stampante è stata correttamente aperta essa è pronta per stampare testi. Dobbiamo allora dare in modo immediato l'istruzione PRINT per stampare i testi specifici.

Controllate allora che la stampante sia accesa, che abbia il nastro e che vi sia la carta. Date le istruzioni OPEN e CMD:

```
OPEN 1,4:CMD 1
```

Al comando RETURN la testina andrà a capo e si posizionerà a sinistra. Potete allora stampare qualcosa per prova, ma sempre tra virgolette:

```
PRINT "CARLO ALBERTO"
```

Al comando RETURN il cursore sparisce dallo schermo e la stampante batte:

```
READY
CARLO ALBERTO
```

Quando stampate in modo immediato, la prima riga porta il messaggio READY mentre sulla successiva appare il vostro testo. Al termine la testina va a capo e riappare il cursore sullo schermo. A questo punto potete stampare altre righe.

Ecco alcuni altri esempi:

Come listare un programma sulla stampante

Per listare un programma su stampante battete i comandi di apertura OPEN e CMD e poi semplicemente LIST:

```
OPEN 1,4:CMD 1
LIST
```

LIST può ovviamente essere integrato con i suoi parametri di linea. Il vostro programma PROVA può essere così stampato:

```
OPEN 1,4:CMD 1
LIST

 10 FOR I=1 TO 800
 20 PRINT "A";
 30 NEXT I
 40 PRINT UFFA!
 50 END
READY.
#
```

Il comando di chiusura CLOSE

La linea logica tra calcolatore e stampante deve essere chiusa quando si sono finite le operazioni di stampa. Il comando per questa chiusura è CLOSE:

```
CLOSE 1
```

Il numero dopo CLOSE deve essere lo stesso della corrispondente OPEN. Per esempio:

```
OPEN 1,4          OPEN 15,4

.                 .
CLOSE 1           CLOSE 15
```

Attenzione però che prima di chiudere dovete aver dato almeno un ordine PRINT # per chiudere correttamente. Ecco alcuni esempi validi ed errati:

Valido		Errato

`OPEN 5,4`
`PRINT#5,"BUON GIORNO"`
`CLOSE 5`		

`OPEN 5,4`
`CMD 5,"BUON GIORNO"`
`PRINT#5:CLOSE 5`
	no	`OPEN 5,4`
		`CMD 5,"BUON GIORNO"`
		`CLOSE 5`

`OPEN 5,4`
`CMD 5,"BUON GIORNO"`
`PRINT#5,"BUON GIORNO"`
`CLOSE 5`
	no	`OPEN 5,4`
		`CMD 5,"BUON GIORNO"`
		`PRINT#5,"BUON GIORNO"`
		`PRINT#5:CLOSE 5`

`OPEN 5,4`
`PRINT#5,"BUON GIORNO"`
`CMD 5,"BUON GIORNO"`
`PRINT#5:CLOSE 5`
	no	`OPEN 5,4`
		`PRINT#5,"BUON GIORNO"`
		`CMD 5,"BUON GIORNO"`
		`CLOSE 5`

CAPITOLO 3

ELABORAZIONE DI TESTI "EDITING"

I calcolatori CBM permettono di visualizzare sullo schermo i caratteri così come sono battuti sulla tastiera. È possibile inoltre elaborare o modificare ("editing") in modo immediato qualunque testo appaia sullo schermo.

L'elaborazione dei testi sullo schermo o "editing" è una delle risorse più importanti dei calcolatori CBM. Potete modificare con estrema semplicità ed efficienza il vostro testo sullo schermo mediante l'editor che descriviamo in questo capitolo. Se trovate delle difficoltà a comprendere quanto spieghiamo adesso leggete prima il capitolo 4 "Programmazione dei calcolatori CBL" e poi ritornate a questo capitolo. Nel capitolo 4 vi daremo infatti i concetti fondamentali per programmare nel linguaggio BASIC.

Il programma editor permette di muovere sullo schermo il cursore in quattro direzioni e i caratteri possono essere cancellati, modificati o inseriti in qualunque posizione.

I comandi del cursore sono già stati descritti nel capitolo 1 al paragrafo "Gruppi di tasti delle tastiere CBM". Il tasto CLEAR SCREEN/HOME muove il cursore alla prima posizione in alto a sinistra e può anche cancellare tutto lo schermo. I tasti CURSOR UP/DOWN e CURSOR LEFT/RIGHT spostano il cursore verso l'alto o il basso e verso sinistra o destra. Il tasto INSERT/DELETE permette di inserire o cancellare un singolo carattere.

ELABORAZIONE DEL TESTO SULLA LINEA CORRENTE

Spesso, mentre inserite una riga di testo, potete accorgervi di aver commesso un errore. È possibile correggere subito l'errore con i comandi CURSOR LEFT e DELETE.

Facciamo un esempio per illustrare come funziona l'"editing" su una riga battendo questo testo:

```
MIO CARO ANICO
```

ANICO è ovviamente errato e vogliamo quindi cambiare la N con M.

Con il tasto CURSOR LEFT spostiamo il cursore all'indietro verso sinistra senza modificare quanto già scritto. Con il tasto DELETE cancelliamo invece, sempre all'indietro, tutto il testo sino al carattere errato N. La scelta tra un metodo e l'altro dipende dalla lunghezza della riga e di quanti caratteri verso sinistra dovreste

spostarvi. È ovvio che, usando, DELETE, dovete ribattere tutti i caratteri a destra del punto di correzione con il pericolo quindi di commettere nuovi errori. In tal caso conviene usare il comando CURSOR LEFT.

Spostamento all'indietro con il tasto CURSOR LEFT

Battete questo testo senza premere RETURN:

```
MIO CARO ANICO▒
```

Il cursore deve essere spostato all'indietro verso la N che sarà cambiata in M così che ANICO divenga AMICO. Per fare questo premete SHIFT e contemporaneamente CURSOR LEFT tante volte quante è necessario per portare il cursore sopra la lettera N.

Se avete un calcolatore CBM 8000 il cursore si muove automaticamente per tutto il tempo che tenete premuto il suo tasto a differenza delle altre tastiere ove è necessario premere e rilasciare ripetutamente.

```
MIO CARO ANICO▒. ←— Premere CURSOR LEFT
MIO CARO ANICO ←— Premere CURSOR LEFT
MIO CARO ANICO ←— Premere CURSOR LEFT
MIO CARO ANICO ←— Premere CURSOR LEFT
MIO CARO ANICO
```

Una volta che il cursore è sopra la N battete una M che subito sostituirà la vecchia lettera mentre il cursore si sposterà di un carattere a destra. Questo modo di operare si dice appunto "per sovrapposizione". Sullo schermo avremo quindi il nostro testo corretto:

```
MIO CARO AMICO
```

Potete ora riportare il cursore a destra premendo il tasto CURSOR RIGHT (senza SHIFT) e battere un'ulteriore parte della riga oppure, se avete finito, battere RETURN. Provate in ambedue i casi.

Fate però attenzione che **solo con il comando RETURN una riga che avete battuto e corretto viene accettata dalla memoria del calcolatore. Se per caso avete mosso il cursore con i comandi HOME, CLEAR SCREEN o CURSOR UP/DOWN potete essere tratti in inganno perchè vedete la riga corretta sullo schermo, ma essa non è stata ancora registrata nel calcolatore.** Ricordate quindi che per confermare una riga dovete sempre battere RETURN.

Spostamento all'indietro con il tasto DELETE

Caricate ancora l'esempio di prima senza premere RETURN:

```
MIO CARO ANICO▒
```

Premete DELETE cancellando all'indietro il testo fino alla lettera errata.

Quando il cursore si posizionerà nel punto ove prima c'era la lettera errata N, battete allora la M.

```
MIO CARO ANICO▒ ← Premere DELETE
MIO CARO ANIC▒ ← Premere DELETE
MIO CARO ANI▒ ← Premere DELETE
MIO CARO AN▒ ← Premere DELETE
MIO CARO A▒
```

Per ripristinare il testo dovete ribattere i caratteri che avete cancellato, ma attenzione a non fare più errori di quanti appena corretti. Alla fine premete RETURN per rendere permanenti in memoria i cambiamenti fatti.

```
MIO CARO A▒ ← Battere M
MIO CARO AM▒ ← Battere I
MIO CARO AMI▒ ← Battere C
MIO CARO AMIC▒ ← Battere O
MIO CARO AMICO▒
```

Spostamento e cancellazione con il tasto DELETE

Il tasto DELETE può essere usato anche per spostare di una posizione verso sinistra un testo e contemporaneamente cancellare il carattere posto alla sinistra del cursore.

Supponiamo di avere battuto due volte la I:

```
MIO CARO AMIICO▒
```

Dobbiamo quindi cancellare una I e compattare la parola AMICO. Supponiamo di cancellare la prima delle due I. Invece di cancellare il testo sino alla prima I muoviamo il cursore verso sinistra, con il tasto CURSOR LEFT, fino alla seconda I.

```
MIO CARO AMIICO
```

Battiamo allora DELETE che per l'appunto cancella la lettera immediatamente alla sua sinistra. Con questo comando si ottiene anche che tutto il testo alla destra si sposti verso sinistra di un carattere, riempiendo quindi lo spazio vuoto che sarebbe rimasto.

```
MIO CARO AMIICO ← Posizione del cursore prima di DELETE
MIO CARO AMICO ← Dopo DELETE
```

Per confermare questa riga di testo potete prima spostare il cursore a destra con CURSOR RIGHT e poi battere RETURN, oppure battere subito RETURN.

Spostamento del testo con il tasto INSERT

Il tasto INSERT apre uno spazio nel testo nella posizione in cui è il cursore e sposta, di un carattere verso destra, tutta la parte della riga posta dopo il cursore.

Per mostrare quanto detto sopra provate a battere un testo tralasciando un carattere:

```
MIO CARO AMCO
```

la I che manca può essere inserita in vari modi.

Potete usare il tasto DELETE e cancellare tutto il testo sino alla I che manca e poi ribattere tutto come abbiamo già visto precedentemente. Questo metodo è sconsigliabile se si deve cancellare più di qualche carattere, perchè è facile fare nuovi errori nella fase di ribattitura del testo. È meglio usare allora il tasto INSERT che non richiede cancellature.

Prima di toccare il tasto INSERT, muovete il cursore verso sinistra fin sopra la posizione dove dovete inserite il nuovo carattere e cioè sul carattere che sarà contiguo a destra a quello da inserire.

```
MIO CARO AMCO ← Posizione del cursore prima di INSERT
              └── Inserire una I
```

Nel nostro esempio con il tasto CURSOR LEFT spostate il cursore sulla lettera C:

```
MIO CARO AMCO ← Premere CURSOR LEFT
MIO CARO AMCO ← Premere CURSOR LEFT
MIO CARO AMCO
```

Premete allora INSERT una volta così che apparirà uno spazio tra la M e la C mentre le lettere CO si sposteranno a destra. Il cursore invece rimarrà fermo.

```
MIO CARO AM CO ← Dopo aver battuto INSERT
```

Ora potete battere la I:

```
MIO CARO AMICO ← Battere I
```

Per chiudere questa riga potete infine premere RETURN subito o dopo aver spostato il cursore verso destra.

Se invece di un solo carattere dobbiamo inserire in un testo una intera parola dovremo battere più volte INSERT così da creare lo spazio sufficiente per la nuova parola. Proviamo per esempio a battere:

```
SAN CAMPANARO
```

e ci accorgiamo di aver dimenticato MARTINO.

```
SAN        CAMPANARO
    ‿‿‿‿‿‿‿‿‿‿‿‿‿‿‿‿
    MARTINO
```

Con il tasto CURSOR LEFT spostiamo allora il cursore sino alla C:

```
SAN CAMPANARO▒
SAN CAMPANARO
SAN CAMPANARO
SAN CAMPANARO
SAN CAMPANARO
SAN CAMPANARO
SAN CAMPANARO
SAN CAMPANARO
SAN CAMPANARO
SAN CAMPANARO
```

Premiamo quindi INSERT otto volte per fare posto alla parola MARTINO più uno spazio:

```
SAN CAMPANARO  <------ Premere INSERT
SAN ▒CAMPANARO  <------ Premere INSERT
SAN ▒ CAMPANARO  <------ Premere INSERT
SAN ▒  CAMPANARO  <------ Premere INSERT
SAN ▒   CAMPANARO
SAN ▒    CAMPANARO
SAN ▒     CAMPANARO
SAN ▒      CAMPANARO
SAN ▒       CAMPANARO
```

E infine battiamo la parola MARTINO:

```
SAN ▒        CAMPANARO
SAN M▒       CAMPANARO
SAN MA▒      CAMPANARO
SAN MAR▒     CAMPANARO
SAN MART▒    CAMPANARO
SAN MARTI▒   CAMPANARO
SAN MARTIN▒  CAMPANARO
SAN MARTINO▒CAMPANARO
```

Per confermare il testo premiamo RETURN.

Ci sono alcune regole da ricordare quando si lavora con il trasto INSERT: muovere sempre il cursore fino dove l'inserimento deve iniziare. Il carattere sotto il cursore si sposterà a destra diventando il primo carattere dopo l'inserimento

Quando inserite più caratteri premere tante volte INSERT quanti sono i caratteri da inserire più gli spazi.

ELABORAZIONE DI TESTI RACCHIUSI TRA VIRGOLETTE

Nel caso che il testo sia racchiuso tra virgolette, la sua elaborazione o "editing" dovrà avvenire con procedure diverse in quanto le virgolette stesse sono interpretate dal calcolatore come l'inizio e la fine di una stringa.

Come abbiamo già detto nel capitolo 2, ogni testo battuto dopo un numero dispari di virgolette viene riconosciuto come una stringa. Questo vale anche per un comando dato al cursore che apparirà sullo schermo con un simbolo speciale, ma non muoverà il cursore. Una stringa prima di essere elaborata, deve essere chiusa con le seconde virgolette oppure premendo direttamente RETURN. Una volta fuori dalla stringa i tasti del cursore operano normalmente. **Nei calcolatori CBM 8000 il comando ESC cancella gli effetti di un numero dispari di virgolette.**

Diamo un esempio di come operare all'interno di una stringa:

```
PRINT "MIO CARO AMICO"
```

Questa è una istruzione di stampa che farà apparire sullo schermo la stringa di caratteri MIO CARO AMICO quando si prema RETURN.

Supponiamo, anche in questo caso, di aver commesso un errore:

```
PRINT "MIO CARO AMIXO▒
```

ove appunto si è battuto X invece di C.

Per correggere l'errore potremmo pensare di spostare il cursore con il tasto CURSOR LEFT, ma ci accorgeremmo subito che invece di far muovere il cursore vedremmo apparire sullo schermo un carattere speciale incorporato nella stringa. Solo quando l'istruzione PRINT sarà eseguita i comandi di cursore avranno effetto e sposteranno il cursore.

```
PRINT "MIO CARO AMIXO▌▌▒ ←— Premuto 2 volte
```

Nella Tabella 3-1 sono illustrati i simboli speciali che rappresentano i comandi di cursore in una stringa.

È possibile evitare di "programmare" i comandi di cursore in una stringa chiudendola con le sue virgolette prima di iniziare l'editing.

Ritorniamo al nostro esempio; prima di toccare i comandi del cursore completiamo il testo della stringa e chiudiamo le virgolette:

```
PRINT "MIO CARO AMIXO"
```

Tabella 3-1: Rappresentazione dei tasti di cursore nelle stringhe

Funzione		Tasto	Simbolo nella stringa
DELETE		INST DEL	(Reverse shift T)
INSERT	Shift	INST DEL	
Home Cursor		CLR SCREEN HOME	(Reverse S)
Clear Screen	Shift	CLR SCREEN HOME	(Reverse Shift S)
Cursor Down		CURSOR ⇧⇩	(Reverse Q)
Cursor Up	Shift	CURSOR ⇧⇩	(Reverse Shift Q)
Cursor Right		CURSOR ⇦⇨	(Reverse])
Cursor Left	Shift	CURSOR ⇦⇨	(Reverse Shift])

A questo punto possiamo usare il cursore come abbiamo visto in precedenza perchè non siamo più dentro la stringa.

```
PRINT "MIO CARO AMIXO"%  ← Premere CURSOR LEFT
PRINT "MIO CARO AMIXO"   ← Premere CURSOR LEFT
PRINT "MIO CARO AMIXO"   ← Premere CURSOR LEFT
PRINT "MIO CARO AMIXO"   ← Premere CURSOR LEFT
```

Battete allora la C sopra alla X e terminate la riga con RETURN. Sullo schermo appare infine la stringa corretta:

```
PRINT "MIO CARO AMIXO"   ← Battere c
PRINT "MIO CARO AMICO"   ← Premere RETURN
MIO CARO AMICO           ← Esecuzione della PRINT

READY
%
```

Esiste un altro metodo per corregere l'interno di una stringa. Battete RETURN come se la stringa fosse corretta e per il momento non preoccupatevi di come il calcolatore risponde ai vostri comandi. A noi interessa per ora correggere il contenuto della stringa. Al comando RETURN quindi il cursore va a capo dopo READY. Spostate allora verso l'alto il cursore con il tasto CURSOR UP e poi verso destra con CURSOR RIGHT. Proviamo nel nostro esempio:

```
PRINT "MIO CARO AMIXO"
```

Premiamo RETURN:

```
PRINT "MIO CARO AMIXO"   ← Premere RETURN
MIO CARO AMIXO

READY
%
```

Ignoriamo la risposta sullo schermo e premiamo CURSOR UP varie volte sino alla riga originaria.

```
Premere CURSOR UP   → PRINT "MIO CARO AMIXO"
Premere CURSOR UP   → MIO CARO AMIXO
Premere CURSOR UP   →
Premere CURSOR UP   → READY
                      %
```

Portiamo ora il cursore con CURSOR RIGHT fin sopra la X e finalmente battiamo la C.

```
PRINT "MIO CARO AMIXO"
PRINT "MIO CARO AMICO"
```

Da ultimo premete RETURN e di conseguenza avrete sullo schermo la vostra stringa corretta:

```
PRINT "MIO CARO AMICO"
MIO CARO AMICO           ← RETURN premuto

READY
%
```

I comandi CURSOR LEFT, CURSOR RIGHT e CURSOR UP/DOWN se dati all'interno di una stringa fanno parte della stringa stessa.

I comandi INSERT e DELETE si comportano invece diversamente.

Il comando INSERT entra a far parte di una stringa come qualunque altro carattere con il simbolo (◫). Ma quando il comando PRINT è eseguito e la stringa visualizzata, INSERT non ha più effetto sullo schermo.

Il tasto DELETE è l'unico comando del cursore che non risenta della presenza delle virgolette. Possiamo quindi correggere il nostro testo usando DELETE come abbiamo già illustrato in precedenza.

```
PRINT "MIO CARO AMINO"▒ ←— Premere DELETE
PRINT "MIO CARO AMINO▒ ←— Premere DELETE
PRINT "MIO CARO AMI▒ ←— Premere DELETE
PRINT "MIO CARO AMI▒ ←— Premere DELETE
```

ELABORAZIONE (EDITING) DELLE RIGHE DI UN PROGRAMMA

È preferibile che rimandiate la lettura di questo paragrafo sino a che non conoscerete la programmazione BASIC.

In questo paragrafo spieghiamo come duplicare righe di programma simili modificando il loro numero di riga.

Duplicazione delle righe

Spesso in un programma vi sono righe simili o identiche per cui può essere molto utile duplicare una riga iniziale invece di ribatterla varie volte. Ricordiamo che ogni riga di programma, nei calcolatori CBM, deve avere un suo diverso numero. **Basterà quindi cambiare il numero di riga per avere una nuova riga senza doverla ribattere.**

Carichiamo la seguente riga di programma:

```
10 PRINT "*"
```

Supponiamo di aver bisogno di altre cinque righe eguali a questa. Ovviamente potremmo ribatterla cinque volte con una diversa numerazione, ma invece vi mostriamo come è possibile, solo cambiando il numero di riga, ottenere le nostre nuove righe.

Battete allora la riga originale (se il programma è già in memoria allora richiamate la vostra riga semplicemente con il comando LIST seguito dal numero di quella riga). Portate con il tasto CURSOR UP il cursore sopra la cifra iniziale del numero di riga. Battete ora sopra il vecchio numero il primo dei nuovi:

```
20 PRINT "*" ←— Battere il numero 20 sopra al 10
```

Appena una nuova riga è pronta memorizzatela con RETURN. Se provate a listare il programma vedrete che ambedue le righe sono presenti:

```
 20 PRINT "*"
LIST

 10 PRINT "*"
 20 PRINT "*"
READY.
```

Proseguite così per formare tutte le nuove righe.

Elaborazione di righe simili

Per elaborare righe di programma simili procediamo in maniera analoga a quanto visto per duplicare righe eguali.

Listate la riga da duplicare specificando il suo numero e portate il cursore al suo inizio. Battete il nuovo numero di riga e quindi fate le modifiche necessarie con i comandi che già conoscete CURSOR RIGHT, INSERT o DELETE. Premete infine RETURN in modo che la nuova riga vada in memoria. Non preoccupatevi se non vedete più sullo schermo la vecchia riga perchè essa è sempre in memoria e potete leggerla in qualunque momento con un comando di LIST.

Per maggior chiarezza facciamo un esempio. Supponiamo di voler scrivere un programma come questo:

```
10 PRINT "*"
20 PRINT " *"
30 PRINT "  *"
40 PRINT "   *"
50 PRINT "    *"
```

Dal momento che tutte le righe sono simili duplicheremo la prima quattro volte.

Carichiamo la riga 10 e premiamo RETURN. Per creare la riga numero 20 spostiamo il cursore in alto sopra il 10 e battiamo 20. Poi spostiamo il cursore a destra con CURSOR RIGHT sino sopra l'asterisco (*) e premiamo INSERT per creare uno spazio e muovere il resto della riga verso destra. Con RETURN confermiamo infine la nuova riga. Sebbene la riga numero 10 non sia più presente sullo schermo essa è sempre in memoria e può essere richiamata, come già detto, con il comando LIST:

```
 20 PRINT " *"<-- Istruzione 20 creata dalla 10
LIST

 10 PRINT "*"
 20 PRINT " *"
READY.
```

Le righe 30, 40 e 50 saranno scritte nella stessa maniera della 20 muovendo prima il

cursore in alto e poi modificando sia il numero di riga che il testo. Alla fine sullo schermo dovrà rimanervi:

```
50 PRINT "    *"
```

Controlliamo infine che tutto sia memorizzato correttamente listando il programma:

```
50 PRINT "    *"
LIST

 10 PRINT "*"
 20 PRINT " *"
 30 PRINT "  *"
 40 PRINT "   *"
 50 PRINT "    *"
READY.
```

Come avete visto righe simili possono essere facilmente duplicate ed elaborate con l'editing.

SVILUPPI DELL'ELABORAZIONE DEI TESTI NEL BASIC 4.0

A differenza delle versioni precedenti, nel BASIC 4.0 sono previste delle ulteriori possibilità di editing. Tali facilitazioni sono generalmente impiegate all'interno di un programma e quindi non si usano in modo immediato. Esse saranno descritte nel capitolo 5.

CAPITOLO 4

PROGRAMMAZIONE DEI CALCOLATORI CBM

In questo capitolo iniziamo a descrivere il linguaggio BASIC.

Il BASIC è un linguaggio di programmazione e come qualunque altro linguaggio è composto da un insieme di istruzioni. Una successione finita di tali istruzioni costituisce un programma. La scelta e l'ordine con cui vengono scritte le istruzioni dipende dai compiti che volete assegnare al vostro calcolatore.

Per insegnarvi a programmare in BASIC potremmo, a questo punto, darvi la spiegazione di ogni istruzione una alla volta. Ma questo metodo sarebbe molto tedioso e otterremmo solo che voi apprendiate molte regole di sintassi senza darvi metodi pratici di programmazione.

Abbiamo così rinviato al capitolo 8 la definizione rigorosa di ogni istruzione BASIC dei calcolatori CBM; in questo capitolo cerchiamo invece di mostrarvi, con esempi pratici, come affrontare la stesura di un programma.

MODI DI PROGRAMMAZIONE: IMMEDIATA E DIFFERITA

Un calcolatore CBM appena viene acceso si pone nello stato di programmazione detto "modo immediato". In tale modo di lavoro esso funziona come una calcolatrice tradizionale e cioè esegue subito le istruzioni BASIC appena premete il tasto RETURN. Provate questi esempi aritmetici:

```
?4.5+6.42                 Addizione
 10.92

READY.
?500-410                  Sottrazione
 90

READY.
?π*2                      Moltiplicazione
 6.28318531

READY.
?100/3                    Divisione
 33.3333333

READY.
?6/2*4-1                  Espressione
 11
```

Vedrete infatti che il risultato viene visualizzato subito sulla riga successiva a quella dell'istruzione.

Nel secondo modo di lavoro detto "differito" o "a programma" il calcolatore prima riceve tutte le istruzioni e poi le esegue solo se gli date il comando di esecuzione RUN ("gira!", "via!").

Programmi e istruzioni

I cinque esempi di istruzioni in modo immediato dati sopra sono in realtà cinque mini-programmi. Più in generale un programma è composto da molte istruzioni così da eseguire un lavoro intero e ben definito (si dice che i calcolatori implementano l'esecuzione di algoritmi).

Negli esempi di sopra i programmi sono costituiti da una sola istruzione, ma normalmente un programma comprende decine e anche centinaia di istruzioni.

Esecuzione di un programma

Si dice che un calcolatore esegue un programma (talvolta in gergo si dice "fare il run" oppure "lanciare" un programma) quando svolge le operazioni che sono previste dal programma stesso.

Un programma in modo immediato viene eseguito appena premuto il tasto RETURN.

Un programma in modo differito viene invece eseguito quando si sia data la particolare istruzione RUN (vedi capitolo 1).

Linee di programma

Se lavorate in modo differito ogni linea (o riga) di programma ha un suo proprio numero; se questo numero manca allora il calcolatore presume che la linea d'istruzione sia data in modo immediato.

Una linea di programma può essere lunga sino a 80 caratteri.

Se avete un display con schermo a 80 colonne su una sua riga potrete visualizzare una linea intera di programma. Se invece lo schermo ha 40 colonne una linea di programma sarà riportata su due righe.

Se una linea del vostro programma è più corta di 80 caratteri allora essa ha termine quando premete il tasto RETURN. Talvolta è consentito andare oltre gli 80 caratteri, **ma per sicurezza terminate sempre le vostre linee di programma prima dell'80-esimo carattere con il comando RETURN.**

Sia in modo immediato che in quello differito una linea di programma può contenere più di una istruzione purchè in totale essa non sia lunga più di 80 caratteri.

PROGRAMMI CON UNA SOLA LINEA IN MODO IMMEDIATO

Un programma in modo immediato deve stare in una unica linea poichè esso viene eseguito appena si preme il tasto RETURN. Esso può contenere però più di una

istruzione per cui si possono scrivere dei programmi in modo immediato molto interessanti. Vediamo alcuni esempi.

Abbiamo già visto che l'istruzione PRINT può essere abbreviata battendo il carattere punto interrogativo. E sappiamo anche che tale istruzione fa apparire un testo sullo schermo del display. Consideriamo allora queste due istruzioni in modo immediato:

```
A=π*2

READY.
?A
 6.28318531
```

La prima A = π * 2 non fa apparire niente sullo schermo perchè non contiene l'istruzione ?, ma fa eseguire un calcolo all'interno del calcolatore. la seconda istruzione fa apparire il risultato sullo schermo.

Quando più istruzioni sono scritte su un'unica linea, esse devono essere separate dal carattere due punti (:).

Così le due istruzioni:

```
A=π*2
?A
```

possono essere condensate in un'unica linea:

```
A=π*2:?A
```

che può essere considerata come un programma dato in modo immediato.

Siccome una linea può contenere fino a 80 caratteri è facile immaginare quante istruzioni essa può contenere. Per esempio considerate la linea seguente:

```
FOR I=1 TO 800: ?"A";: NEXT: ?"UFFA!"
```

Anche se non vi è possibile per ora comprenderne esattamente il significato provate a batterla sulla vosta tastiera. Appena batterete poi il comando di fine riga RETURN vedrete apparire sullo schermo 20 righe di 40 caratteri (se lo schermo è a 40 colonne) tutte riempite con la lettera A e alla 21-esima riga l'esclamazione UFFA!

```
FOR I=1 TO 300: ?"A": NEXT: ?"UFFA!"
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
UFFA!

READY.
*
```

La linea di programma rimane visibile perchè in totale vengono occupare esattamente le 25 righe dello schermo. Se avete invece un display con schermo a 80 colonne la lettera A apparirà su 10 righe lunghe 80 caratteri e la linea di programma sarà sempre visualizzata sopra le dieci righe di lettere A.

Ri-esecuzione di un programma in modo immediato

Al termine dell'esecuzione di una linea di programma in modo immediato appare il messaggio READY mentre il cursore ritorna all'inizio della riga successiva.

Se ora desiderate eseguire nuovamente la stessa linea è sufficiente che voi riportiate il cursore all'estremo destro di questa linea e ribattiate RETURN. **Con il BASIC CBM voi potete infatti rieseguire una qualunque linea purchè essa sia presente sullo schermo.** Per poter fare questo basta che spostiate il cursore, con i suoi comandi, al termine della riga e poi battiate RETURN. Mentre spostate il cursore potete anche modificare il testo della linea come in una qualunque operazione di editing.

Nel nostro esempio provate a premere HOME per portare il cursore in alto a sinistra poi spostatelo a destra di 15 caratteri sopra la A e battete un altro carattere come per esempio il DIAGONAL QUARTER-BLOCK SOLID (tasto ? shiftato). Poi battete RETURN e vedrete così visualizzare sullo schermo il nuovo carattere.

```
FOR I=1 TO 800: ?"▚";: NEXT: ?"UFFA!"
▚▚▚
```

6. Per togliere le due virgolette che sono rimaste, spostate a destra il cursore con CURSOR RIGHT

```
C$="W": FOR I=1 TO 800: PC$": NEXT: ?"UFFA!"
```

e poi battete DELETE:

```
C$="W": FOR I=1 TO 800: PC$: NEXT: ?"UFFA!"
```

Dopo tutti questi cambiamenti avete la nuova riga e per eseguirla battete RETURN. Se volete cambiare il carattere che viene visualizzato premete HOME e portate il cursore sul vecchio carattere; battete il nuovo carattere e infine RETURN.

Le spaziature in una linea non sono necessarie

Non preoccupatevi di porre molta attenzione alle spaziature in una riga! **L'interprete BASIC CBM riconosce gli elementi di un'istruzione anche se sono attaccati tra loro.** Per esempio:

```
120 FOR I=1 TO 210
```

può essere scritta:

```
120 FOR I=1 TO210
```

oppure anche:

```
120 FORI=1TO210
```

Analogamente potete aggiungere qualunque ulteriore spaziatura, ma non all'interno di parole proprie del linguaggio BASIC. Potete scrivere GOTO oppure GO TO, ma non potete scrivere F OR per FOR. Le uniche spaziature che mantengono inalterata la loro posizione sono quelle poste in una stringa tra virgolette. In altre parole si può dire che le spaziature poste in una istruzione hanno lo scopo fondamentale di rendere l'istruzione stessa più facilmente comprensibile.

ELEMENTI DI UN LINGUAGGIO DI PROGRAMMAZIONE

Le istruzioni di un programma devono essere scritte rispettando un insieme ben definito di regole che prende il nome di "sintassi".

Ogni linguaggio di programmazione ha la sua specifica sintassi ed è così possibile definire un gran numero di linguaggi di programmazione. **I calcolatori CBM impiegano un unico linguaggio residente, detto BASIC, e tutte le regole che vi descriviamo in questo libro costituiscono appunto la sintassi del BASIC CBM.**

I linguaggi di programmazione assomigliano, in un certo senso, alle lingue parlate e anche tra loro si possono individuare alcuni linguaggi più "parlati" come il FORTRAN, il COBOL, il BASIC, il PASCAL, l'APL, il PL/M, il PL-1 e il FORTH. Forse in totale, compresi anche i linguaggi meno noti, essi sono più di un centinaio.

Purtroppo i linguaggi di programmazione assomigliano alle lingue parlate anche per le derivazioni"dialettali". **Un programma scritto in BASIC, per un calcolatore CBM, non girerà mai direttamente su un altro calcolatore non CBM anche se quest'ultimo opera in BASIC.** Infatti vi saranno delle piccole differenze di sintassi che non permetteranno mai l'interscambio diretto dei programmi. È opportuno però dirvi subito che una volta imparato un linguaggio specifico è molto facile imparare anche gli altri "dialetti".

Alcune regole di sintassi sono ovvie. Gli esempi con addizione e sottrazione all'inizio di questo capitolo non richiedono infatti alcuna spiegazione. Altre regole sono invece assolutamente arbitrarie per cui non dovete cercare una spiegazione, ma solo imparare la loro sintassi. Per esempio perchè usare "★" per indicare la moltiplicazione invece del tradizionale segno "x" ? Semplicemente perchè il calcolatore non potrebbe distinguere tra la lettere "x" e il segno di moltiplicazione. E così in quasi tutti i linguaggi di programmazione si è scelto l'asterisco "★" per indicare la moltiplicazione. La divisione è invece universalmente rappresentata dal segno "/". L'unico motivo perchè non si è scelto il simbolo "÷" è che questo segno non è presente sulle tastiere dei calcolatori e delle telescriventi.

Iniziamo ora la descrizione delle sintassi del BASIC CBM che possiamo ripartire in tre capitoli principali: sintassi della numerazione delle righe, dei dati e dei comandi al calcolatore.

NUMERI DI LINEA

Come abbiamo già avuto occasione di dire, **un programma scritto in modo differito deve avere un numero all'inizio di ogni linea.** La prima linea deve avere il numero più basso, l'ultima il numero più alto e tutte le altre numeri progressivi crescenti. **Potete battere le linee con qualunque ordine perchè il calcolatore stesso le porrà in ordine crescente.** Consideriamo per esempio un vecchio programma che abbia i seguenti numeri di linea:

```
120
130
140
150
160
170
180
190
```

Se ora caricate una nuova linea con il numero 165 essa sarà inizialmente posta per ultima, ma il calcolatore provvederà automaticamente a porla tra la 160 e la 170.

Questo spostamento può essere così illustrato:

Numeri di linea al momento che battete la linea 165	*Inserimento automatico della nuova linea*
120	120
130	130
140	140
150	150
160	160
170	165
180	170
190	180
	190
165	

Se una nuova linea ha il numero di una già esistente allora la vecchia linea viene completamente sostituita da quella nuova.

I numeri di linea del BASIC CBM devono essere compresi tra 1 e 63999. Il calcolatore interpreta le eventuali cifre poste all'inizio di una linea come numero di linea e se per caso avete battuto un numero con più di cinque cifre vi verrà subito fatto notare che avete commesso un errore di sintassi.

Tutti i "dialetti" BASIC richiedono una numerazione delle linee in ordine crescente sebbene il valore massimo consentito dipenda da ogni singolo linguaggio.

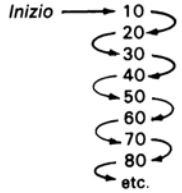
Gli altri linguaggi di programmazione, diversi dal BASIC, non richiedono la numerazione di tutte le linee e spesso non richiedono neanche che eventuali numeri siano in ordine crescente.

I numeri di linea hanno anche l'importante significato di indirizzo. **Infatti le istruzioni di un qualunque linguaggio si possono dividere in due grandi gruppi:**

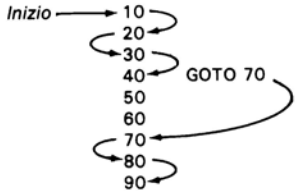
1. Istruzioni che creano o modificano dei dati.

2. Istruzioni che stabiliscono e controllano la successione delle operazioni.

Il concetto che le operazioni di un programma debbano essere eseguite con un certo ordine è facilmente comprensibile. Spesso le istruzioni sono eseguite con lo stesso ordine sequenziale con cui sono scritte e cioè per esempio:



Ma altre volte è necessario effettuare dei salti per cui il numero di linea ci permette appunto di indirizzare il salto. Questo può essere illustrato così:



DATI

Le istruzioni di un programma hanno lo scopo fondamentale di stabilire come operare sui dati e altre volte di attribuire loro un valore concreto. Vi descriviamo ora i vari tipi di dati che potete usare nel BASIC CBM.

Per quanto riguarda i numeri è consuetudine dividerli in due famiglie: numeri con virgola (detti anche frazionari o reali o "floating point") e numeri interi.

Numeri con virgola (o frazionari)

La numerazione con virgola è la rappresentazione standard dei numeri nei calcolatori CBM. Tutte le operazioni aritmetiche sono infatti eseguite su numeri con virgola. Un numero con virgola può essere privo della parte decimale e corrispondere quindi ad un numero intero. Può essere negativo (−) o positivo (con o senza il segno +). Ecco alcuni esempi di numeri con virgola che corrispondono a numeri interi:

```
5
-15
65000
161
0
```

mentre questi sono numeri con virgola che hanno anche una parte decimale:

```
0.5
0.0165432
-0.0000009
1.6
24.0055
-64.2
3.1416
```

Fate ben attenzione che anche se diciamo "numeri con virgola" in realtà noi dobbiamo porre un punto per rispettare la grafia anglosassone. Se poniamo la virgola commettiamo un errore di sintassi. Inoltre non dobbiamo neanche porre il punto per separare le migliaia, i milioni, ecc.; per esempio dobbiamo scrivere 65000 e non 65.000.

Arrotondamento

Tutti i numeri hanno almeno otto cifre significative, ma alcuni possono averne anche nove. Se voi tentate di aggiungere altre cifre il BASIC CBM effettua una operazione di arrotondamento. Se la nuova cifra è maggiore o eguale a cinque si arrotonda per eccesso, mentre se è eguale o minore a quattro si arrotonda per difetto. Vi sono però

alcune eccezioni come quelle riportate in questi esempi:

```
?.5555555556
 .555555555

?.5555555557
 .555555556

?.1111111115
 .111111111

?.1111111116
 .111111112
```

Sembra che arrotondi da 6 in giù e da 7 in su

Sembra che arrotondi da 5 in giù e da 6 in su

Rappresentazione scientifica

Un numero con dieci o più cifre viene sempre convertito nella forma di rappresentazione detta scientifica. Per esempio:

```
READY.
?1111111114
1.11111111E+09

READY.
?1111111115
1.11111112E+09
```

Un numero nella rappresentazione scientifica ha la forma:

numero E + ee

dove:

numero	può essere un numero intero, frazionario o una espressione. Esso rappresenta le cifre significative che sono anche chiamate "coefficiente". Se non appare il punto, è implicito che esso sia alla destra del coefficiente.
E	equivale a "esponente".
+	può essere il segno + o quello −.
ee	sono le due cifre dell'esponente decimale. Si può anche dire che rappresentano il numero di posti di cui spostare il punto a sinistra (esponente negativo) o a destra (esponente positivo) per portarlo nella sua posizione effettiva.

Ecco alcuni esempi:

Rappresentazione scientifica	Rappresentazione standard
2E1	20
10.5E+4	105000
66E+2	6600
66E−2	0.66
−66E−2	−0.66
1E−10	0.0000000001
94E20	9400000000000000000000

La rappresentazione scientica dei numeri è molto utile per esprimere numeri molto piccoli oppure molto grandi. Il BASIC CBM rappresenta infatti nella forma standard i numeri compresi tra 0.01 e 999999999 mentre usa invece quella scientifica per i numeri fuori da tale intervallo. Ecco ancora qualche esempio:

```
?.009
 9E-03

READY.
?.01
 .01

READY.
?999999998.9
 999999999

READY.
?999999999.6
 1E+09
```

È importante precisare che anche con la rappresentazione scientifica esiste un limite alla grandezza dei numeri che si possono elaborare con il BASIC CBM. Questi limiti sono:

Il più grande numero con virgola è: + 1.70141183E + 38
Il più piccolo numero con virgola è: + 2.93873588E − 39

Ogni numero di grandezza maggiore darà un errore di "overflow" (superamento). Per esempio:

```
?1.70141183E+38
 1.70141183E+38
READY.
?-1.70141183E+38
-1.70141183E+38
```

Nessun errore di overflow

```
READY.
?1.70141184E+38

?OVERFLOW ERROR
READY.
?-1.70141184E+38
```

Errore di overflow

```
?OVERFLOW ERROR
```

Mentre numeri più piccoli, del numero più piccolo possibile, saranno convertiti in zero:

```
?2.93873588E-39
 2.93873588E-39

READY.
?-2.93873588E-39
-2.93873588E-39
```

Questi numeri sono corretti

```
READY.
?2.93873587E-39
 0

READY.
?-2.93873587E-39
 0
```

Numeri troppo piccoli: sono sostituiti da zero

Numeri interi

Un numero è intero quando non ha la parte decimale o frazionaria. Può essere negativo (−) o positivo (con o senza il segno +). Nei calcolatori CBM un numero intero è compreso nell'intervallo tra −32767 e +32767. Ecco alcuni esempi:

```
0
1
44
32699
-15
```

Ogni numero intero può essere rappresentato come numero con virgola poichè i numeri interi sono un sottoinsieme dei numeri frazionari. **Le operazioni aritmetiche sui numeri interi sono eseguite convertendo dapprima i numeri interi in numeri frazionari.**

Stringhe

La parola "stringa" è usata per indicare dati che non sono numeri bensì parole come per esempio le parole o le frasi di questo libro.

Abbiamo già usato la parola stringa per indicare i testi che appaiono come messaggi sullo schermo del display. **Una stringa è costituita da uno o più caratteri racchiusi tra virgolette;** per esempio:

```
"HAI!"
"SINERGIA"
"1234567"
"IL TOTALE È 34,54"
"MILANO, VIA CARCANO 63"
```

Tra le virgolette di una stringa potete inserire caratteri alfabetici o numerici, simboli o caratteri grafici, caratteri di controllo del cursore (CLEAR SCREEN,

HOME CURSOR, CURSOR UP/DOWN, CURSOR LEFT/RIGHT) e il tasto REVERSE ON/OFF. **Gli unici tasti che non possono essere inseriti in una stringa sono RUN/STOP, RETURN e INSERT/DELETE.**

Se una stringa viene visualizzata tutti i suoi caratteri appaiono così come sono. I tasti di controllo del cursore e REVERSE ON/OFF non comportano alcuna stampa se battuti per cui si è dovuto associarli a qualche simbolo per renderli riconoscibili.

Vedete infatti a questo scopo la Tabella 4-1.

Siccome una stringa viene normalmente definita all'interno di un'istruzione, essa deve essere lunga meno di 80 caratteri perchè alcune posizioni sono occupate dal numero di linea e dalle virgolette.

In un calcolatore CBM si possono memorizzare però stringhe lunghe sino a 255 caratteri concatenando stringhe più corte. Di questo ne parleremo in seguito.

Variabili

Quando vi abbiamo parlato della programmazione in modo immediato vi abbiamo fatto l'esempio delle due istruzioni:

```
A=π*2
?A
```

che potevano essere condensate in un'unica istruzione:

```
A=π*2 ?A
```

In questi esempi A rappresenta il nome di una variabile.

Una variabile è un modo convenzionale per rappresentare un numero o una stringa. Per esempio:

```
100 A=B+C
200 ?A
```

Questo piccolo programma fa stampare la somma di due numeri, ma esso potrà essere eseguito solo se si danno valori concreti alle due variabili B e C. Infatti se assegnamo a B e a C due valori:

```
90 B=4.65
95 C=3.72
100 A=B+C
200 ?A
```

Otteremmo A = 8.37.

Tabella 4-1: Simboli speciali nelle stringhe

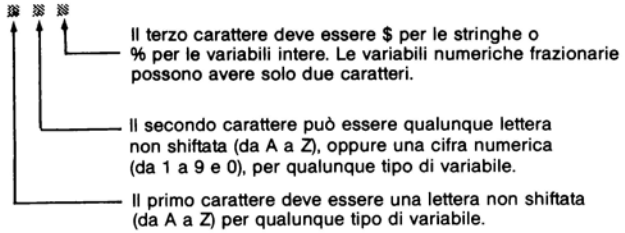
Funzione		Tasto	Simbolo nella stringa*
Reverse On		OFF RVS ON	(Reverse R)
Reverse Off	Shift	OFF RVS ON	(Reverse Shift R)
Home Cursor		CLR SCREEN HOME	(Reverse S)
Clear Screen	Shift	CLR SCREEN HOME	(Reverse Shift S)
Cursor Down		CURSOR	(Reverse Q)
Cursor Up	Shift	CURSOR	(Reverse Shift Q)
Cursor Right		CURSOR	(Reverse])
Cursor Left	Shift	CURSOR	(Reverse Shift])

* Il simbolo grafico che appare in questa colonna può variare tra un calcolatore CBM ad un altro a seconda del tipo di tastiera. La descrizione dei tasti è però comunque valida.

I nomi di variabili possono essere usati sia per indicare dati numerici che stringhe. Se avete avuto occasione di studiare l'Algebra pensiamo vi sia facile capire che cosa sono le variabili. Diversamente provate a pensare alle variabili come al nome di una casella postale: ogni cosa sia inserita nella casella diviene il valore assegnato alla variabile.

Nomi di variabili

Il nome di una variabile può essere costituito da uno, due o tre caratteri con le seguenti condizioni:



Si vede così che l'ultimo carattere caratterizza il tipo di variabile.

Prestate molta attenzione perchè i primi due caratteri vogliono lettere o numeri con il tasto SHIFT non premuto. In tal caso potrete avere lettere minuscole o maiuscole a seconda dell'insieme di caratteri in funzione in quel momento, **ma quello che conta in ogni caso è che SHIFT non sia abilitato!**

Le variabili per numeri con virgola sono le più usate, eccone alcuni esempi:

```
A
B
C
A1
AA
Z5
```

Nel caso di numeri interi avete invece variabili come queste:

```
A%
B%
C%
A1%
MN%
X4%
```

Ricordate che anche le variabili per numeri con virgola possono assumere valori interi. E infine ecco alcuni esempi di variabili di stringa:

```
A$
M$
MN$
M1$
ZX$
F6$
```

Se lo ritenete opportuno potete scrivere nomi di variabili lunghi sino a 255 caratteri, ma solamente i primi due caratteri sono utili per il riconoscimento della variabile. Per

esempio BANANA e BACIO sono la stessa variabile (BA). Altri esempi sono:

MAGIC$	*interpretato come*	MA$
N123456789	*interpretato come*	N1
MMM$	*interpretato come*	MM$
ABCDEF%	*interpretato come*	AB%
CALENDAR	*interpretato come*	CA

Se desiderate usare nomi di variabile più lunghi di due caratteri ricordatevi però che:

1. Nomi più lunghi occupano più spazio in memoria però permettono una più facile lettura del programma e delle sue variabili.
2. Solo i primi due caratteri (e il simbolo $ o %) identificano la variabile.
3. Nel BASIC CBM esistono delle parole riservate, riportate nella Tabella 4 (a pag. 120), che non possono assolutamente far parte dei nomi delle variabili.

OPERATORI

L'istruzione BASIC:

```
100 ?10.2+4.7
```

ordina al calcolatore di sommare 10.2 a 4.7 e di visualizzare il risultato. Invece l'istruzione:

```
250 C=A+B
```

ordina al calcolatore di sommare due numeri con virgola rappresentati dalle variabili A e B e di assegnare il risultato alla variabile C.

Come è facile intuire il segno (+) indica l'operazione di addizione. Nella terminologia del calcolatori si dice appunto che il segno (+) rappresenta un operatore aritmetico.

Oltre all'operatore somma, tutti noi conosciamo gli operatori differenza, moltiplicazione e divisione. Esistono però altre classi di operatori che sono molto facili da capire, ma che normalmente non usiamo nella nostra attività quotidiana: gli operatori relazionali e gli operatori Booleani.

In Tabella 4-2 riportiamo tutti gli operatori del BASIC CBM descritti dettagliatamente nei paragrafi successivi.

Operatori Aritmetici

Gli operatori aritmetici trattano le operazioni di somma, sottrazione, moltiplicazione, divisione ed elevazione a potenza. Essi operano su numeri con virgola. Come abbiamo già detto, se le operazioni sono fatte su numeri interi questi vengono

Tabella 4-2: Operatori

	Precedenza	Operatore	Significato
	Alta 9	( )	Le parentesi cambiano l'ordine di precedenza
Operatori Aritmetici	8	↑	Potenza
	7	–	Segno meno
	6	*	Moltiplicazione
	6	/	Divisione
	5	+	Addizione
	5	–	Sottrazione
Operatori Relazionali	4	=	Eguale
	4	< >	Non eguale
	4	<	Minore di
	4	>	Maggiore di
	4	< = o = <	Minore o uguale di
	4	> = o = >	Maggiore o eguale di
Operatori Booleani	3	NOT	Negazione logica
	2	AND	AND logico
	1	OR	OR logico
	Bassa		

dapprima trasformati in numeri con virgola, poi viene eseguita l'operazione. Se il risultato deve essere intero allora esso viene convertito dalla forma con virgola a quella intera.

I dati su cui agiscono gli operatori sono detti "operandi".

Gli operatori aritmetici operano sempre su due operandi.

Gli operandi possono essere numeri oppure variabili.

Addizione (+). Il dato o operando alla sinistra del segno più viene sommato all'operando di destra. Per esempio:

```
2+2
A+B+C
X%+1
BR+10E-2
```

Il segno (+) viene anche usato per sommare stringhe, ma con la grande differenza che invece di sommare due valori effettua invece il concatenamento delle stringhe.

La differenza tra somma di due numeri e somma di due stringhe può essere facilmente compresa osservando questi esempi:

```
Addizione di numeri:
  numero 1 + numero 2 = numero 3

Addizione di stringhe:
  stringa 1 + stringa 2 = stringa 1 stringa 2
```

Mediante il concatenamento di più stringhe si può costruire una stringa lunga sino a 255 caratteri; per esempio:

"FOR" + "WARD"	risulta: "FORWARD"
"BUON" +" "+" GIORNO"	risulta: "BUON GIORNO"
A$ + B$	risulta il concatenamento delle due stringhe contenute in A$ e B$
"1" + CH$	risulta il concatenamento del carattere "1" e del contenuto di CH$

In questo esempio se A$ è eguale a "FOR" e B$ è eguale a "WARD" allora A$ + B$ porta allo stesso risultato di "FOR" + "WARD".

Sottrazione (−). L'operando di destra viene sottratto all'operando di sinistra. Per esempio:

4 − 1	risultato: 3
100 − 64	risultato: 36
A − B	risulta la differenza tra il contenuto della variabile A e della variabile B
55 − 142	risultato: −87

In questo esempio se ad A viene assegnato il valore 100 e a B il valore 64 allora la seconda e terza riga portano allo stesso risultato.

Il segno meno viene anche usato per indicare numeri negativi. Per esempio:

−5	
−9E4	
−B	
4−−2	Equivale a 4 + 2

Moltiplicazione (★). Il segno di asterisco indica che l'operando alla sua destra viene moltiplicato per l'operando alla sua sinistra:

100★2	risultato: 200
50★0	risultato: 0
A★X1	risulta il prodotto tra i due numeri frazionari rappresentati dalle variabili A e X1

Se alla variabile A viena assegnato il valore 4.2 e alla variabile X1 il valore 9.63, il risultato della moltiplicazione sarà 40.446. A e X1 potrebbero anche contenere numeri interi come 100 e 2 ma essi sarebbero posti nella forma con virgola, cioè

100.0 e 2.0, perchè le due variabili sono appunto variabili numeriche con virgola. Se volessimo invece che i due valori siano rappresentati come interi dovremmo aggiungere il carattere % alle variabili: A% e X1%.

Divisione (/). L'operando alla sinistra della barra è il dividendo mentre quello alla destra è il divisore. Esempio:

10/2	risultato: 5
6400/4	risultato: 1600
A/B	risulta il quoziente tra i due numeri frazionari contenuti nelle variabili A e B
4E2/XR	il numero 400 viene diviso per il contenuto di XR

Il terzo esempio rappresenta un generico quoziente purchè si assegnino ad A e a B opportuni valori. Se vogliamo che la divisione avvenga solo tra numeri interi dovremo aggiungere il segno %: A% / B%.

Elevazione a potenza (↑). Il dato alla sinistra della freccetta viene elevato alla potenza indicata dal dato posto alla destra. Se il secondo dato o operando è 2 si dice che si eleva al quadrato; se è 3 si dice che si eleva al cubo. Il secondo operando, detto anche esponente, può essere un numero o una variabile o una espressione purchè il risultato dell'operazione porti ad un valore di grandezza non superiore a quella massima consentita per i numeri con virgola nel calcolatore CBM.

2↑2	risultato: 4
12↑2	risultato: 144
1↑3	risultato: 1
A↑5	il valore della variabile frazionaria A viene elevato alla potenza 5
2↑6.4	risultato: 84.4485064
NM↑– 10	il valore della variabile NM viene elevato alla potenza –10
14↑F	il numero 14 viene elevato alla potenza rappresentata dal valore di F

Ordine di esecuzione delle operazioni aritmetiche

Una espressione aritmetica può contenere più operazioni come illustrato in questo esempio:

```
A+C*10/2↑2
```

Quando il calcolatore incontra una espressione come questa esegue le singole

operazione con un ben determinato ordine: **prima di tutto calcola le potenze (↑), poi tiene conto dei segni, poi esegue le moltiplicazioni e le divisioni (* /) e infine esegue le addizioni e le sottrazioni** (+ −). Le operazioni dello stesso livello gerarchico sono eseguite nell'ordine da sinistra verso destra.

Se usate però le parentesi, come normalmente fate in Algebra, potete stabilire un qualunque ordine di esecuzione. Ecco un esempio di come, con le parentesi, potete cambiare l'ordine di esecuzione e ottenere risultati diversi:

```
4+1•2          Risultato: 6
(4+1)•2        Risultato: 10
100•4/2−1      Risultato: 199
100•(4/2−1)    Risultato: 100
100•(4/(2−1))  Risultato: 400
```

Se in una espressione sono presenti coppie di parentesi, una dentro l'altra, il calcolatore inizierà ad operare da quella più interna verso l'esterno. L'uso di parentesi è sempre consigliabile, sia per evitare errori di calcolo, sia per rendere più leggibile il vostro programma.

Operatori relazionali

Gli operatori relazionali rappresentano le seguenti condizioni: maggiore di (>), minore di (<), eguale (=), diverso da (< >), maggiore o eguale di (> =) e minore o eguale di (< =).

```
1 = 5−4        risultato: vero (−1)
14 > 66        risultato: falso (0)
15 > = 15      risultato: vero (−1)
A < > B        il risultato dipende dai valori
                 assegnati alle variabili A e B
```

Il BASIC CBM assegna il valore 0 ad una condizione "falsa" e il valore −1 ad una condizione "vera". Questi due numeri 0 e −1 possono essere usati in una qualunque espressione aritmetica. Per esempio l'espressione (1 = 1) * 4 diverrà −4 perchè la condizione (1 = 1) è "vera" e quindi vale −1. In altre parole, potete sempre inserire in una espressione operatori relazionali. Ecco alcuni esempi:

```
25+(14>66)                Equivale a: 25+0
(A+(1=5−4))•(15>=15)      Equivale a: (A−1)·(−1)
```

Gli operatori relazionali possono essere usati anche per confrontare stringhe. La relazione d'ordine che viene scelta in questo caso è quella alfabetica e cioè: A < B, B < C, C < D, ecc. Due stringhe sono confrontate carattere per carattere a cominciare da quello più a sinistra. Per esempio:

```
"A" < "B"          risultato: vero (−1)
"X" = "XX"         risultato: falso (0)
C$ = A$ + B$       il risultato dipende dal valore
                     assegnato alle tre variabili di stringa
                     C$, A$ e B$
```

Il BASIC CBM assegna il valore −1 per una relazione "vera" tra due stringhe e il valore 0 per una relazione "falsa" così come viene fatto per le relazioni tra numeri. Per esempio:

```
("JONES" > "DOE")+37                      Equivale a: −1+37
("AAA" < "AA")•(Z9−("OTTER" > "AB"))      Equivale a: 0·(Z9−(−1))
```

Operatori Booleani

Gli operatori Booleani vi permettono di prendere decisioni logiche nell'ambito del vostro programma. Comunemente questo operatori sono quattro: AND, OR, OR ESCLUSIVO e NOT, ma **nel BASIC CBM sono previsti solo tre di questi operatori: AND, OR e NOT.**

Se non conoscete già gli operatori Booleani, o come più esattamente si dice l'Algebra Booleana, allora cerchiamo di darvene un'idea con un esempio.

Supponiamo che dovete comprare dei biscotti per due bambini in un negozio. L'operatore AND dice che comprerete i biscotti se il bambino A e il bambino B desiderano i biscotti (AND in inglese significa "e").

L'operatore OR dice che i biscotti saranno comperati se il bambino A oppure il bambino B li desiderano (OR in inglese significa "o"). In questo caso ricordatevi però che li possono desiderare anche tutti e due!

L'operatore NOT genera semplicemente una negazione. Se il bambino A non è mai d'accordo con il bambino B allora diremo che la decisione di A è la negazione di quella di B.

I calcolatori numerici non lavorano però con analogie, ma bensì con numeri. Di conseguenza la logica Booleana riduce tutte le variabili ai soli valori 0 e 1. **Nella Tabella 4-3 è riportata la cosidetta "tabella della verità" che indica tutti i possibili valori che possono essere assunti dalle variabili logiche.**

Gli operatori Booleani possono essere usati per controllare la successione logica delle istruzioni di un programma:

```
IF A = 100 AND B = 100 GOTO 10
  Se ambedue A e B sono eguali a 100,
  il programma salta alla linea 10

IF X < Y AND B >= 44 THEN F = 0
  Se X è minore di Y, e B è maggiore o eguale
  a 44, allora F viene posta eguale a 0

IF A = 100 OR B = 100 GOTO 20
  Se una delle due variabili A o B è uguale a 100
  allora il programma salta alla linea 20

IF X < Y OR B >= 44 THEN F = 0
  F è posto eguale a 0, se X è minore di Y
  oppure se B è maggiore o eguale a 44

IF A = 1 AND B = 2 OR C = 3 GOTO 30
  il salto alla linea 30 avviene se A = 1 e
  contemporaneamente B = 2, oppure se C
  è uguale a 3
```

È possibile verificare se un singolo operando è "vero" o "falso". Se un operando appare da solo è implicito che sia seguito dalla diseguaglianza "<>0". Ogni valore non nullo è considerato vero mentre ogni valore nullo è ritenuto falso.

```
IF A THEN B = 2
IF A <> 0 THEN B = -2
  queste due istruzioni sono equivalenti

IF NOT B GOTO 100
  il salto avviene se B è falso.
  È preferibile però scrivere:
  IF B = 0 GOTO 100
```

Tutti gli operatori Booleani usano operandi interi. Se applicate un operatore Booleano ad un numero con virgola allora tale valore sarà automaticamente convertito in intero. È ovvio che questa seconda operazione sarà possibile solo se al numero con virgola iniziale corrisponde un intero di grandezza non superiore alla massima prevista per i numeri interi nel vostro calcolatore.

Nel caso di variabili di stringa non è assolutamente consentito applicare loro operatori Booleani.

Se siete un programmatore principiante forse avrete qualche difficoltà per comprendere i paragrafi successivi; in tal caso saltate pure alla prossima sezione.

Tabella 4-3: Tavola della verità Booleana

Il risultato dell'operazione AND vale 1 solo se i due bit sono 1

```
1 AND 1 = 1
0 AND 1 = 0
1 AND 0 = 0
0 AND 0 = 0
```

Il risultato dell'operazione OR vale 1 se uno o ambedue i bit sono 1

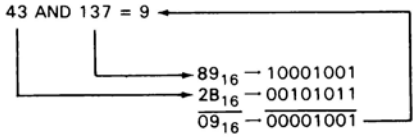
```
1 OR 1 = 1
0 OR 1 = 1
1 OR 0 = 1
0 OR 0 = 0
```

Il risultato dell'operazione NOT è il complemento del bit

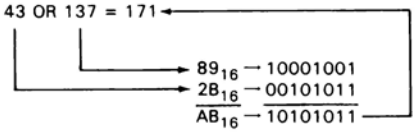
```
NOT 1 = 0
NOT 0 = 1
```

Gli operatori Booleani operano su operandi interi una cifra binaria alla volta. Il BASIC CBM memorizza tutti i numeri in forma binaria usando la notazione di complemento a due per i valori negativi. Possiamo allora illustrare una operazione

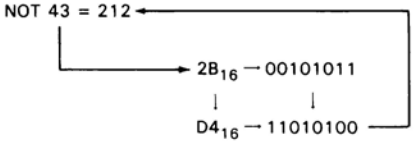
AND in questo modo:

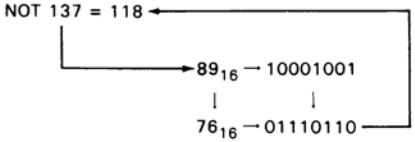


Una operazione OR invece:



E da ultimo due operazioni NOT:





Operazioni Booleane di questo tipo sono normalmente impiegate nella applicazioni industriali.*

Se gli operandi non sono interi, sono dapprima convertiti nella forma intera.

Se gli operandi di una operazione Booleana sono relazionali allora assumono i valori 0 o —1. Per esempio:

```
A=1 OR C<2
```

è equivalente a:

$$\left\{\begin{matrix}-1\\0\\0\end{matrix}\right\} \text{OR} \left\{\begin{matrix}-1\\0\\0\end{matrix}\right\}$$

Consideriamo una operzione più complessa:

```
IF A=B AND C<D GOTO 40
```

* Se desiderate approfondire lo studio dell'aritmetica binaria e delle operazioni Booleane potete leggere "An Introduction to Microcomputers: Volume 0 – The Beginners Book" di A. Osborne edito da McGraw-Hill, nel 1977.

Dapprima si devono calcolare le sue espressioni relazionali. Supponiamo che la prima sia vera e la seconda falsa per cui la nostra operazione può essere così riscritta:

```
IF -1 AND 0 GOTO 40
```

Se eseguiamo ora l'operazione AND otteniamo il valore 0:

```
IF 0 GOTO 40
```

Ricordiamo che un singolo termine è sempre seguito da "<>0", per cui otteniamo:

```
IF 0<>0 GOTO 40
```

che è sicuramente falsa per cui il salto alla linea 40 non viene mai fatto.

Diversamente un operatore Booleano, applicato a due variabili numeriche, può portare a qualunque numero intero:

```
IF A% AND B% GOTO 40
```

Poniamo A% = 255 e B% = 240. L'operazione 255 AND 240 porta al risultato 240. Per cui possiamo scrivere:

```
IF 240 GOTO 40
```

oppure:

```
IF 240 <> 0 GOTO 40
```

E in tal caso avverrà il salto alla linea 40.

Confrontiamo ora le due istruzioni di assegnazione:

```
A = A AND 10
A = A <10
```

Nel primo caso il valore corrente di A è congiunto (AND) con il valore 10 e il risultato diviene il nuovo valore di A. Ovviamente A doveva avere inizialmente un valore compreso tra −32767 e +32767. Nel secondo caso l'espressione relazionale A < 10 potrà assumere i valori 0 o −1 per cui ad A potranno essere assegnati solo uno di questi due valori.

VARIABILI CON INDICI ("ARRAYS")

Le variabili con indici (o variabili vettoriali o matriciali) hanno un importantissimo impiego nella programmazione. Qui di seguito cerchiamo di illustrarvi i concetti fondamentali con dei semplici esempi.

Quando avete due o più dati, tra cui esista una correlazione comune, invece di dare ad ognuno di essi un nome diverso potete attribuire a tutti uno stesso nome di variabile e poi distinguere ogni singolo componente con un indice numerico.

Lo scontrino di un supermercato potrebbe, per esempio avere sei voci dal reparto carni e pollame, quattro da quello frutta e verdura e tre da quello latte e formaggi. Le voci di ogni gruppo potrebbero essere chiamate con lo stesso nome; per esempio così:

```
CP$(0)="BOLLITO"
CP$(1)="FILETTO"
CP$(2)="CONIGLIO"
CP$(3)="BISTECCHE"
CP$(4)="POLLAME"
CP$(5)="SALUMI"
```

```
FV$(0)="LIMONI"
FV$(1)="MELE"
FV$(2)="BANANE"
FV$(3)="CAROTE"
```

```
LF$(0)="LATTE"
LF$(1)="PARMIGGIANO"
LF$(2)="CREMA"
```

Ogni singola voce sarebbe poi riconosciuta da un diverso indice.

È possibile estendere ulteriormente questo concetto introducendo un secondo indice e attribuendo a tutte le voci lo stesso nome. Il primo indice indicherebbe quindi il reparto mentre il secondo avrebbe lo stesso significato che aveva nell'esempio precedente:

```
SM$(0,0)=CP$(0)
SM$(0,1)=CP$(1)
SM$(0,2)=CP$(2)
SM$(0,3)=CP$(3)
SM$(0,4)=CP$(4)
SM$(0,5)=CP$(5)
```

```
SM$(1,0)=FV$(0)
SM$(1,1)=FV$(1)
SM$(1,2)=FV$(2)
SM$(1,3)=FV$(3)
```

```
SM$(2,0)=LF$(0)
SM$(2,1)=LF$(1)
SM$(2,2)=LF$(2)
```

Le variabili con indici possono rappresentare sia grandezze intere, che numeri frazionari, che stringhe. **È però assolutamente vietato attribuire ad una stessa variabile con indici contenuti di tipo diverso cioè non si possono mescolare interi con numeri frazionari con stringhe.** Una variabile con indice può rappresentare una sola di queste grandezze!

Le variabili con indici o come più comunemente si dice i vettori, nel caso di un unico indice, o le matrici quando gli indici sono due o più, sono utilissime per denominare un gran numero di variabili che abbiano una qualche caratteristica comune. Considerate per esempio una tabella di 200 numeri posti su 10 righe con 20 numeri per riga (10 righe e 20 colonne) e poniamoci il problema di denominare ognuno dei 200 numeri. È ovvio che sarebbe catastrofico dare ad ogni variabile un nome diverso mentre è facilissimo attribuire un unico nome a tutta la tabella e caratterizzare poi ogni elemento proprio con la coppia di indici che individuano la riga e la colonna.

Le variabili con indici possono avere una o più dimensioni cioè uno o più indici. Se la dimensione è uno, cioè c'è un solo indice, allora si usa dire che abbiamo un vettore. Se la dimensione è due, o superiore a due, si dice più propriamente che abbiamo una

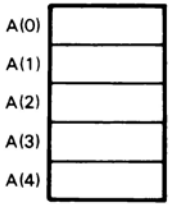
matrice. Nel caso di dimensione due il primo indice individua la riga mentre il secondo individua la colonna. Nel caso di dimensione tre il terzo indice individua piani successivi in cui si pensa scomposta la matrice. Matrice di quattro o più dimensioni sono impossibili da visualizzare per la nostra mente, ma non sono niente di impossibile in matematica.

Vediamo adesso alcune variabili in dettaglio.

Un vettore a una dimensione ha la forma:

```
        nome (i)
dove:
        nome        è il nome della variabile
                      con indice e può essere di
                      qualunque tipo
        i           rappresenta l'indice e inizia
                      da zero
```

Un vettore mono-dimensionale denominato A e con cinque elementi può essere così visualizzato:

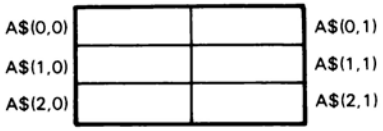


Il numero totale di elementi di un vettore è pari al valore massimo dell'indice più uno in quanto in BASIC si parte dal valore 0 dell'indice.

Una matrice bi-dimensionale ha la forma:

```
        nome (i, j)
dove:
        nome        è il nome della variabile con indici
        i           è l'indice di colonna
        j           è l'indice di riga
```

Per esempio la matrice A$ con due colonne e tre righe può essere così visualizzata:



Si definisce "ordine" di una matrice il prodotto del numero totale di righe per il numero totale di colonne. Nel nostro esempio è 3 x 2 = 6.

Matrici di dimensione superiore possono essere così denominate:

nome (i, j, k, ...)

Le variabili con indice che abbiano fino a 11 elementi, come per esempio un vettore con indice da 0 a 10, possono essere usate correntemente in un programma BASIC. Se il numero totale di elementi è superiore a 11 è necessario "dichiarare" all'inizio del programma la presenza di questa variabile mediante una istruzione di "dimensionamento". Questo tipo di istruzione sarà descritta successivamente. Se date ad una variabile senza indici lo stesso nome di una variabile con indici il calcolatore interpreterà tale variabile come assolutamente distinta da quella con indici.

COMANDI BASIC

Nei capitoli 2 e 3 abbiamo descritto alcuni comandi, che potete dare tramite la tastiera, per controllare lo svolgimento delle funzioni del vostro calcolatore. Vi ricordiamo per esempio il comando RUN.

Tutti i comandi possono essere eseguiti come istruzioni BASIC.

Quando scrivete programmi molto lunghi è probabile che non abbiate sufficiente memoria del calcolatore. Dovete quindi ripartire il vostro programma in tanto moduli separati ed eseguirne uno alla volta. Alcune tecniche, per la gestione di questi moduli di programma, saranno descritte nel capitolo 6.

Parole riservate

Tutte le parole o combinazioni di caratteri che definiscono le istruzioni o funzioni del linguaggio BASIC sono dette "parole riservate". Nella tabella 4-4 abbiamo riportato tutte le parole riservate del BASIC CBM; molte di esse sono state già incontrate e le altre verranno descritte nel capitolo 6.

Il calcolatore, quando esegue un programma, ricerca in ogni istruzione BASIC le stringhe di caratteri che costituiscono le parole riservate. L'unica eccezione è quando la stringa è posta tra virgolette, perchè una stringa viene sempre interpretata come una costante alfanumerica e mai come una istruzione.

Se una parola riservata viene inserita nel nome di una variabile commette un notevole errore. **Ricordatevi di non usare o inserire mai le parole riservate, o le loro abbreviazioni, nei nomi delle variabili.**

L'asterisco posto vicino ad alcune parole nella Tabella 4-4 indica che queste parole appartengono al BASIC versione 4.0 e superiori. È tuttavia consigliabile non usare mai queste parole anche con le altre versioni di BASIC perchè potreste sempre aggiornare il vostro programma per farlo girare su calcolatori con il BASIC 4.0.

Tabella 4-4: Parole riservate.

Parola	Abbreviazioni: Insieme di caratteri alternativo	Abbreviazioni: Insieme di caratteri standard	Parola	Abbreviazioni: Insieme di caratteri alternativo	Abbreviazioni: Insieme di caratteri standard	Parola	Abbreviazioni: Insieme di caratteri alternativo	Abbreviazioni: Insieme di caratteri standard	Parola	Abbreviazioni: Insieme di caratteri alternativo	Abbreviazioni: Insieme di caratteri standard
ABS	aB	A1	DS$*	ds#	DS#	NEW	new	NEW	SCRATCH*	sC	S¬
AND	aN	A	DSAVE*	dS	D♥	NEXT	nE	N¬	SGN	sG	S1
APPEND*	aP	A¬	END	eN	E	NOT	nO	N¬	SIN	sI	S.
ASC	aS	A♥	EXP	e:	E♠	ON	on	ON	SPC(	sP	S¬
ATN	aT	A1	FN	fn	FN	OPEN	oP	O¬	SQR	sQ	S●
BACKUP*	bA	B♠	FOR	fO	F¬	OR	or	OR	ST	st	ST
CHR$	cH	C1	FROM	fR	F=	PEEK	pE	P¬	STATUS	status	STATUS
CLOSE	cl0	CL¬	GET	gE	G	POKE	pO	P¬	STEP	stE	ST¬
CLR	cL	CL	GET #	get#	GET#	POS	pos	POS	STOP	sT	S1
CMD	cM	C	GOTO	gO	G¬	PRINT	?	?	STR$	str#	STR#
COLLECT*	coL	COL	GOSUB	goS	GO♥	PRINT #	pR	P_	SYS	sY	S1
CONCAT*	conC	CON¬	HEADER*	hE	H¬	READ	rE	R¬	TAB(	tA	T♠
CONT	cO	C¬	IF	if	IF	READ #	read#	READ#	TAN	tan	TAN
COPY*	coP	CO¬	INPUT	input	INPUT	RECORD*	reC	RE¬	THEN	tH	T1
COS	cos	COS	INPUT#	iN	I	REM	rem	REM	TI	ti	TI
DATA	dA	D♠	INT	int	INT	RENAME*	reN	RE.	TIME	time	TIME
DCLOSE*	dC	D¬	LEFT$	leF	LE¬	RESTORE	reS	RE♥	TI$	ti#	TI#
DEF	dE	D¬	LEN	len	LEN	RETURN	reT	RE1	TO	to	TO
DIM	dI	D.	LET	lE	L¬	RIGHT$	rI	R.	US	uS	U♥
DIRECTORY*	diR	DI_	LIST	lI	L. .	RND	rN	R	VAL	uA	V♠
DLOAD*	dL	DL	LOAD	lO	L¬	RUN	rU	R.	VERIFY	uE	V¬
DOPEN*	dO	D¬	LOG	log	LOG	SAVE	sA	S♠	WAIT	wA	W♠
DS*	ds	DS	MID$	mI	M.						

* Sono parole riservate nel BASIC versione 4.0 e superiori

Abbreviazioni di parole BASIC

Avete già visto che l'istruzione PRINT può essere inserita nel calcolatore nella forma contratta di ?. Infatti il punto interrogativo ? sarà interpretato dal calcolatore come la parola completa PRINT.

La maggior parte dei comandi, delle istruzioni e delle funzioni BASIC possono essere abbreviate usando i primi due caratteri della parola, ma con il secondo carattere battuto in modo shiftato.

Nel caso di tastiera con caratteri standard il secondo carattere apparirà come un carattere grafico. Per esempio l'abbreviazione di LIST sarà:

Li
c L

Se due parole iniziano con le stesse due lettere (come STEP e STOP) allora l'abbreviazione di due lettere viene assegnata alla parola più usata mentre l'altra o

non viene abbreviata oppure è scritta con tre lettere (di cui la terza in modo shiftato). Per esempio STOP è abbreviato così:

St
o S[illegible]

mentre STEP è abbreviato:

STe
o ST[illegible]

Cioè per abbreviare STEP dovete battere le lettere maiuscole S e T e la E shiftata (che risulta essere il "TOP LINE HORIZONTAL").

Qui di seguito vi diamo alcuni esempi di abbreviazioni con due o tre lettere. **Ricordatevi però che se il programma viene listato esso apparirà nella forma normale espansa.**

```
Po 59468,14          Quindi RETURN (Po abbreviazione per POKE)
10 lE a=10
20 b=a aN 14+eX(2)
30 dI c(5)
40 fO i=0 to 5
50 rE c(I)
60 nE
70 dA 1,6,2,4,10,5,16
80 reS
90 eN
lI                                Abbreviazione per LIST
10 let a=10
20 b=a and 14+exp(2)
30 dim c(5)
40 for i=0 to 5
50 read c(i)
60 next
70 data 1,6,2,4,]
80 restore
90 end
Po 59468,12          Prima di RETURN (Po abbreviazione per POKE)
```

Come abbiamo già detto il programma listato dal calcolatore appare nella forma espansa. Se provate poi a listare, con i caratteri standard, il programma lo vedrete apparire con le lettere maiuscole. Per ritornare ai caratteri standard ricordatevi di dare però il comando:

```
POKE 59468,12
```

Desideriamo a questo punto fare una precisazione che riguarda i comandi SPC e TAB. Questi comandi, quando vengono dati in forma abbreviata, comprendono già la parentesi di sinistra dell'argomento. Per questo motivo se battete così:

```
10 ?sP(5)
```

ottenete in realtà:

```
10 print spc((5)
```

le due parentesi di sinistra
causano un errore di sintassi

che è ovviamente errato.

La forma corretta è questa senza la parentesi di sinistra:

```
10?sP5)
```

Questo comportamento anomalo vale solo per SPC e TAB, per tutte le altre funzioni dovete invece porre ambedue le parentesi.

```
10? rN(1)
```

ISTRUZIONI BASIC

Le operazioni, che si desiderano eseguire con una istruzione, sono indicate mediante le "parole riservate" della Tabella 4-4.

Nel capitolo 8 daremo una descrizione esatta di ogni istruzione BASIC. In questo capitolo diamo invece i concetti fondamentali di programmazione illustrando come si usano le istruzioni. Se lo ritenete opportuno consultate il capitolo 8 per avere l'esatta interpretazione delle singole istruzioni.

COMMENTI

È tradizione che ogni presentazione del linguaggio BASIC cominci con la descrizione dell'unica istruzione che il calcolatore ignora: i "commenti". **Questa istruzione è codificata come REM e ha lo scopo di rendere più facile la lettura del vostro programma mediante l'inserimento di righe che portino vostri commenti o spiegazioni.**

Se scrivete un piccolo programma di cinque o dieci righe non avrete sicuramente difficoltà a ricordare che cosa fa. Ma se è lungo cento o duecento righe, o se avete scritto molti programmi, sarà ben difficile che possiate ricordare la loro strutura. L'unico modo per ovviare a questi inconvenienti è quello di documentare i programmi e cioè inserire in essi delle opportune zone di spiegazione. **Un programmatore esperto riempie i suoi lavori di commenti REM per descrivere in ogni punto la procedura che intende svolgere.**

Le istruzioni REM hanno il numero di linea come qualunque altra istruzione.

ISTRUZIONI DI ASSEGNAZIONE

Le **istruzioni di assegnazione attribuiscono dei valori alle variabili.** Esse sono sicuramente le più usate in un programma. Esempio:

```
100 LET X=3.24
```

L'istruzione 100 assegna il valore 3.24 alla variabile X

```
150 X=3.24
```

Come sopra. L'uso di LET è facoltativo

```
215 A$="BENVENUTI"
```

Alla variabile A$ di stringa viene assegnato il valore "BENVENUTI"

Ecco un altro caso di assegnazione che riguarda il vetore LF$ (I):

```
200 REM  LF$(I) E' LA LISTA DEI LATTICINI
210 LF$(0)="LATTE"
220 LF$(1)="PARMIGGIANO"
230 LF$(2)="CREMA"
```

Se ricordate però che si possono mettere più istruzioni su una stessa riga possiamo allora scrivere:

```
200 REM  LF$(I) E' LA LISTA DEI LATTICINI
210 LF$(0)="LATTE":LF$(1)="PARMIGGIANO":LF$(2)="CREMA"
```

Dove i due punti: servono per separare le istruzioni successive su una stessa riga.

Una istruzione di assegnazione può contenere qualunque operatore aritmetico o relazionale; per esempio:

```
100 V=3.24+7.96/8.5
```

Quest'ultima istruzione assegna il valore 4.17647059 alla variabile V; essa può essere equivalente a queste tre istruzioni:

```
100 X=7.96
110 Y=8.5
120 V=3.24+X/Y
```

che possono essere scritte su una unica riga:

```
100 X=7.96:Y=8.5:V=3.24+X/Y
```

Qui di seguito diamo invece alcuni esempi di assegnazioni che comprendono operazioni Booleane:

```
100 A%=43 AND 137
200 A%=43 OR 137
```

Da ultimo vediamo come una stringa può essere ottenuta mediante il concatenamento di altre stringhe:

```
100 V$="MONTE"
200 W$="CERVINO"
300 D$=V$+" "+W$
```

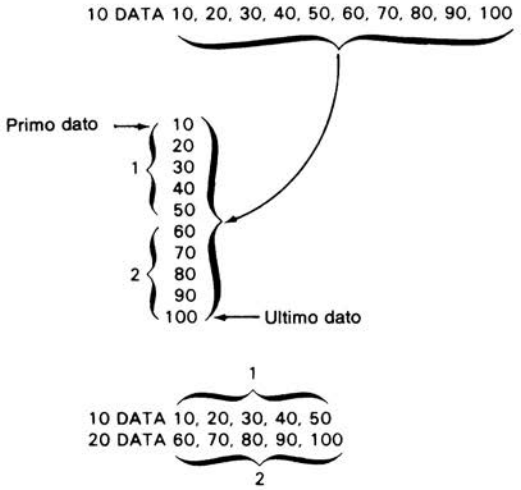
Le istruzioni DATA e READ

Quando le variabili, che necessitano di assegnare loro un valore sono molte, si può ricorrere alla coppia di istruzioni DATA e READ. Considerate questo esempio:

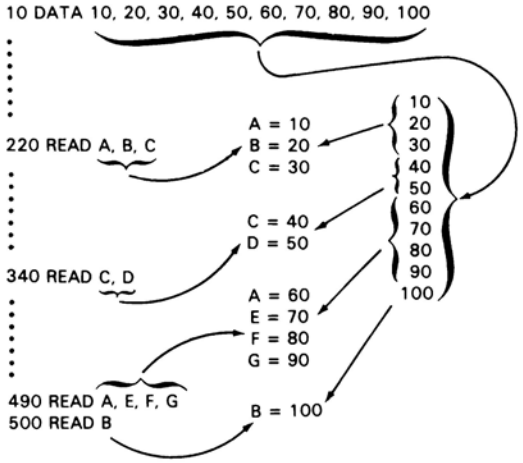
```
10 DATA 10,20,-4,16E5
20 READ A,B,C,D
```

L'istruzione alla riga 10 definisce quattro valori numerici. Questi quattro valori sono assegnati a quattro variabili numeriche con virgola mediante l'istruzione della riga 20. Alla fine avremo: A = 10, B = 20, C = −4 e D = 16 x 10^5.

Se avete una o più istruzioni DATA allora potete pensare che i valori siano posti come in una "colonna". Questa "colonna" di valori può essere costituita da una sola istruzione DATA come anche da più istruzioni successive. In questo disegno cerchiamo di chiarire il concetto:

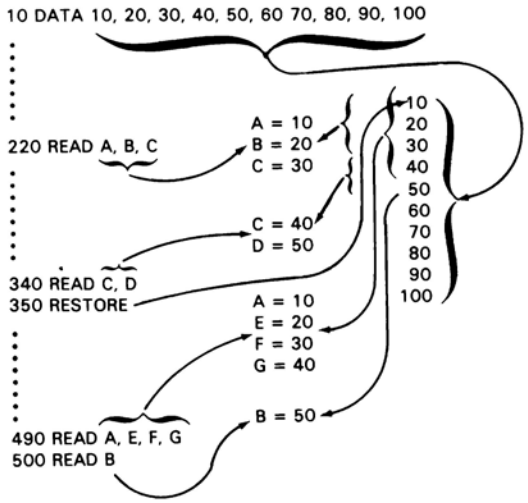


Le istruzioni READ hanno lo scopo invece di assegnare i valori, contenuti nelle istruzioni DATA, alle variabili elencate nella READ stessa. Se sono presenti più di una READ, l'attribuzione dei valori avverrà con il criterio che ogni istruzione READ prende i primi valori lasciati liberi dalla READ precedente. Con questo esempio cerchiamo di dare una illustrazione grafica:



Istruzione RESTORE

Supponiamo esita un puntatore che indichi qual'è il primo valore libero nella colonna di valori della DATA, allora **mediante l'istruzione RESTORE possiamo riportare all'inizio tale puntatore.** In altre parole possiamo utilizzare una seconda volta tutti i valori elencati in DATA. Per esempio:



ISTRUZIONE DI DIMENSIONAMENTO

Se una variabile con indice viene citata in un programma, il BASIC CBM le attribuisce automaticamente la dimensione dieci con possibilità di avere undici elementi. Cioè il suo indice può variare da 0 a 10. **Se volete che la variabile abbia più o meno di undici elementi allora lo dovete dichiarare con una istruzione di "dimensionamento" DIM. Analogamente se avete una matrice con dimensione due o superiore e con qualunque numero di elementi, dovete dichiararla nella stessa istruzione DIM.** Per esempio dimensioniamo alcune variabili che abbiamo già incontrato:

```
DIM CP$(5),FV$(3),LF$(2)

DIM SM$(3,5)
```

Una istruzione DIM può contenente quante si voglia variabili da dimensionare purchè la lunghezza dell'istruzione non superi gli 80 caratteri.

Il numero o i numeri che compaiono tra parentesi dopo le variabili nella DIM devono rappresentare i valori massimi che potranno assumere gli indici. Ricordatevi però che è sempre previsto che il valore più basso di un indice sia 0 per cui il numero effettivo di elementi è pari a quello citato nella DIM più uno. Per esempio CP$ (5) avrà 6 elementi perchè dimensionata con CP$ (5). Analogamente SM$ (3,5) conterrà 24 elementi cioè: (3+1) x (5+1) = 24.

Se durante un programma richiamate una variabile vettoriale o matriciale con una dimensione o un indice non dichiarato nella DIM commetterete un errore di sintassi.

ISTRUZIONI DI SALTO

Normalmente un programma viene eseguito istruzione per istruzione a partire dal numero di riga più basso verso l'alto. Mediante le istruzioni di salto potete cambiare questo ordine di esecuzione.

Istruzione GOTO

GOTO è la più semplice istruzione di salto. **Essa vi permette di stabilire in modo univoco quale sarà l'istruzione successiva.** Considerate per esempio:

```
20 A=4.37
30 GOTO 100
40
50
60
70
80
90
100
110
.
.
```

L'istruzione alla linea 20 è una assegnazione che attribuisce un valore ad A. L'istruzione successiva è un salto che porta l'esecuzione del programma alla linea 100. Possiamo quindi dire che in questa zona del programma le istruzioni saranno eseguite con l'ordine: ... 20, 30, 100 ...

Ovviamente in qualche altro punto del programma esisterà una istruzione di salto alla linea 40 perchè diversamente questa instruzione non sarebbe mai eseguita.

Potete saltare a qualunque linea anche se questa contiene solo i commenti REM. In tal caso il calcolatore prosegue subito alla linea successiva. Per esempio:

```
20 A=4.37
30 GOTO 70
40
50
60
70 REM QUESTO E' UN COMMENTO
80
90
.
.
```

Il programma salta dalla linea 30 alla 70 dove trova un commento per cui va subito alla 80. In questo caso avrete potuto saltare direttamente alla 80.

```
20 A=4.37
30 GOTO 80
40
50
60
70 REM QUESTO E' UN COMMENTO
80
90
.
.
```

Istruzione GOTO calcolato

L'istruzione GOTO calcolato permette di saltare ad una tra diverse linee a seconda che una certa variabile assuma il valore intero 1 o 2 o 3 etc. Per esempio:

```
A%=1   10
       20
       30 A%=B%-2
       40 ON A% GOTO 10,70,150
       50
       60
A%=2   70
       80
       90
       100
       110
A%=3   120
       130
       140
       150
       160
       .
       .
```

L'istruzione alla linea 40 è il nostro ON ... GOTO ... calcolato. Il programma salterà alla linea 10 se A% = 1; alla linea 70 se A% = 2; alla linea 150 se A% = 3. **Se A% dovesse assumere un valore diverso da 1 o 2 o 3 allora avremmo un errore.** In quest'ultimo esempio vi mostriamo come è possibile percorrere in maniera diversa un programma a seconda di come la variabile A% del GOTO assuma valori diversi. Notete che la variabile A% assume il suo valore in funzione di quello di un'altra variabile B%:

```
10 B%=4
20 ?B%
30 A%=B%-2
40 ON A%GOTO 10,70,150
70 ?B%
80 B%=5
90 GOTO 30
150 ?B%
160 B%=3
170 GOTO 20
```

Caricare questo piccolo programma e date il comando RUN. Vedrete apparire una successione di valori.

Per esercizio provate a scrivere un programma che faccia apparire sul video la sequenza: 345345345 ...

ISTRUZIONI DI CONTROLLO A CICLO

Istruzione FOR NEXT

Le istruzioni GOTO e GOTO calcolato vi permettono di programmare una qualunque sequenza logica di istruzioni. **Molto spesso però vi troverete di fronte alla necessità di dover eseguire più volte una o più istruzioni ben definite.** Per esempio se avete un vettore A(I) con 100 elementi e dovete attribuire i valori ai suoi elementi potreste scrivere 100 volte l'istruzione di assegnazione A(I) = valore. Ma questo modo di programmare sarebbe molto faticoso e banale. È molto più elegante allora pensare di eseguire 100 volte la stessa istruzione cambiando ogni volta l'indice I (da 0 a 99) e il valore di attribuzione. **Questo è possibile con le istruzioni di ciclo FOR e NEXT. Per esempio:**

```
10 DIM A(99)
20 FOR I=0 TO 99 STEP 1
30 A(I)=I
40 NEXT I
```

Le istruzioni tra FOR e NEXT saranno eseguite più volte (nell'esempio 100 volte) e così, pur avendo scritto una sola volta A(I) = valore, otteniamo il caricamento di tutto il vettore A.

Per vedere come le cose procedono provate a battere questo programma:

```
10 DIM A(99)
20 FOR I=0 TO 99 STEP 1
30 A(I)=I
35 ?A(I);
40 NEXT I
50 REM SE PONETE UN SALTO GOTO ALLA
60 REM STESSA LINEA, IL CALCOLATORE
70 REM ESEGUE UN CICLO DI ATTESA
80 GOTO 80
```

Come vedete ad ogni passaggio viene visualizzata la variabile A(I). Battete RUN. Vedrete apparire una colonna di 100 numeri da 0 a 99. Per fermare il programma premete STOP.

Le istruzioni tra FOR e NEXT sono eseguite tante volte quanto è specificato nella stessa istruzione FOR. Nel nostro esempio la variabile di controllo del ciclo è I e viene stabilito che deve variare da 0 a 99 con incrementi di 1. Alla prima esecuzione I = 0 per cui avremo:

```
30 A(0)=0
```

Alla seconda volta I viene incrementata del valore indicato dal STEP e cioè 1. Quindi avremo:

```
30 A(1)=1
```

E così via sino a che I raggiunge il valore massimo previsto di 99. Dopo questo il programma prosegue alla linea 50.

Il valore di STEP può essere un qualunque numero intero. Provate a porlo eguale a 5 e rieseguite il programma. In tal caso il ciclo sarà eseguito 20 volte perchè incrementiamo I di 5 ad ogni passaggio. L'ultimo valore eseguito sarà I = 95 perchè al successivo I assumerebbe il valore 100 che supera il massimo previsto di 99. Se vogliamo adesso possiamo spostare il valore massimo e portarlo per esempio a 499: otterremo ancora 100 esecuzioni del ciclo, ma con valori di I e A diversi. Provate a fare queste modifiche ricordando però di cambiare anche il dimensionamento di A.

È possibile anche che lo STEP sia negativo. In tal caso il valore iniziale di I deve essere superiore a quello finale. Per esempio poniamo lo STEP eguale a −1 e riscriviamo il nostro esempio così:

```
10 DIM A(99)
20 FOR I=99 TO 0 STEP -1
30 A(I)=I
35 ?A(I);
40 NEXT I
80 GOTO 80
```

È opportuno a questo punto precisare che i tre valori che si devono dare in una istruzione FOR possono essere inizialmente dati come valori con virgola o addirittura come espressioni. Se sono espressioni, esse vengono calcolate e ridotte quindi ad un valore numerico. Questi valori numerici vengono riportati sempre a valori interi mediante arrotondamento e solo, a questo punto, sono utilizzati come parametri del ciclo FOR NEXT. È molto probabile quindi che a causa dell'arroton-

damento, i vostri parametri non assumano quei valori interi che forse voi avete previsto. **Vi consigliamo quindi di porre nell'istruzione FOR sempre valori interi.** Se questi valori sono il risultato di una espressione è preferibile che la calcoliate separatamente così da essere ben sicuri che i parametri del ciclo assumano valori interi corretti.

Spesso il valore del passo STEP è 1. **In tal caso non è richiesto che scriviate STEP 1 perchè è sottinteso.** Se non scrivete nulla il BASIC CBM assume che il passo sia unitario e positivo. Provate infatti a riscrivere:

```
10 DIM A(99)
15 REM  IL PASSO DEL CICLO E'  UNO
20 FOR I=0 TO 99
30 A(I)=I
35 ?A(I);
40 NEXT I
80 GOTO 80
```

È anche possibile non indicare dopo NEXT la variabile del ciclo, ma è consigliabile farlo sempre per evitare equivoci e rendere il programma facilmente leggibile.

Cicli nidificati (o "a fascio")

Abbiamo già detto che le istruzioni comprese tra FOR e NEXT costituiscono un ciclo perchè sono eseguite più volte di seguito. La presenza di cicli è molto frequente in un programma e spesso anzi è necessario avere un ciclo dentro ad un altro. Le istruzioni all'interno di un ciclo possono essere qualunque e perciò possono essere ancora quelle di FOR e NEXT, necessarie per costituire un altro ciclo interno. Ecco un esempio di nidificazione:

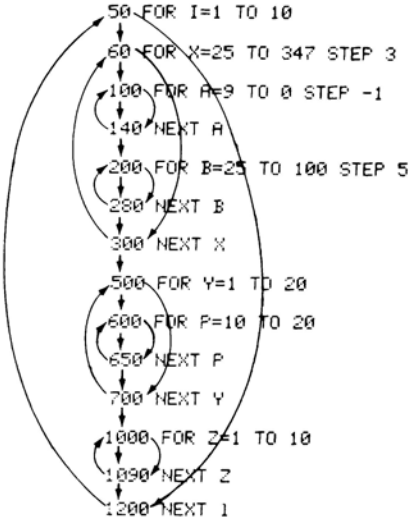
```
10 DIM A(99)
20 FOR I=0 TO 99
30 A(I)=I
40 REM  VISUALIZZA I VALORI DI A(I)
45 REM  APPENA ASSEGNATI
50 FOR J=0 TO I
60 ?A(J)
70 NEXT J
80 NEXT I
90 GOTO 90
```

Srutture complesse di cicli si possono avere anche in programmi moto corti. Ecco un esempio di nidificazione di cicli in cui, per semplicità, non abbiamo riportato le istruzioni intermedie:

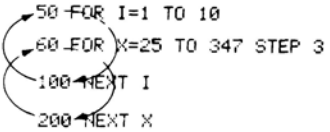
```
50 FOR I=1 TO 10
60 FOR X=25 TO 347 STEP 3
.
100 FOR A=9 TO 0 STEP -1
.
140 NEXT A
200 FOR B=25 TO 100 STEP 5
.
280 NEXT B
300 NEXT X
```

```
500 FOR Y=1 TO 20 STEP 2
.
600 FOR P=10 TO 20
.
650 NEXT P
700 NEXT Y
.
1000 FOR Z=1 TO 10
.
1090 NEXT Z
1200 NEXT I
```

Il ciclo più esterno ha l'indice I. Esso contiene tre cicli tra loro separati con indici X, Y e Z. Il ciclo X a sua volta contiene altri due cicli con indici A e B. Il ciclo Y contiene il ciclo P. **Cicli interni ad un altro devono usare una diversa variabili di ciclo.** L'esecuzione di tale struttura può essere così schematizzata:



L'uso di cicli tra loro nidificati è molto semplice. **L'unico errore che potete commettere, ed è molto grave, è quello di concatenare i cicli cioè di terminare un ciclo più esterno prima di terminare quelli interni.** Per esempio questa struttura è illegale:

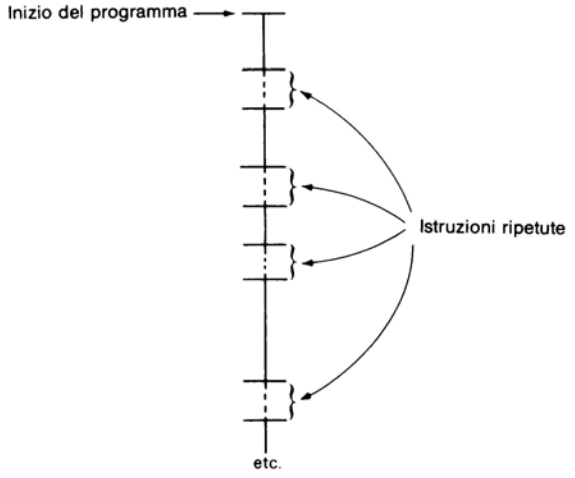


Se non indicate la variabile di ciclo nell'istruzione NEXT il calcolatore provvede da solo a tenerne conto ed esegue correttamente il ciclo. Se fate attenzione, vi accorgerete che esiste una sola possibilità corretta di chiusura dei cicli per cui il BASIC CBM può assegnare automaticamente ad ogni NEXT la sua variabile di ciclo.

È importante precisare che il BASIC CBM vuole un eguale numero di istruzioni FOR e NEXT. Supponete per esempio di avere due cicli uno interno all'altro, ma solo un NEXT. Allora il ciclo più interno sarà eseguito correttamente perchè trova un NEXT, mentre il ciclo esterno non trova un punto terminale. Se invece ponete più NEXT del necessario vi sarà dato un errore di sintassi.

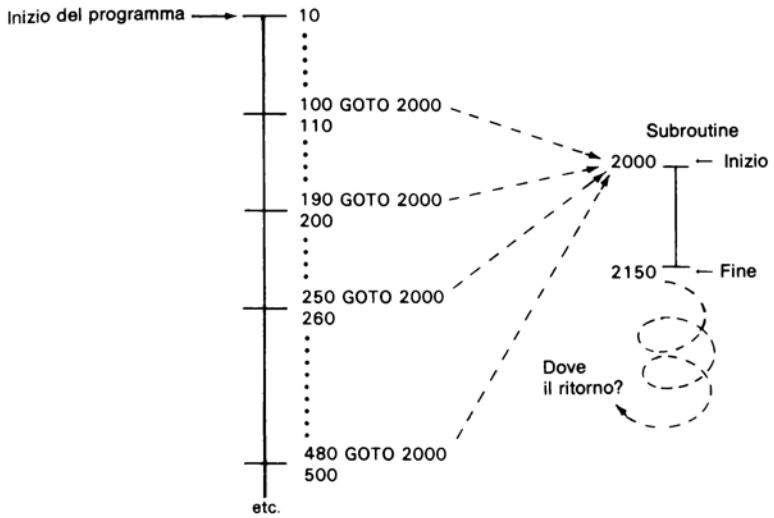
SUBROUTINE (O SOTTOPROGRAMMI)

Può succedere, scrivendo un programma, che una piccola parte di esso debba essere scritta più volte cioè sia necessario eseguire più volte la stessa operazione. Supponete per esempio di avere un vettore A(I) a cui dovete attribuire valori diversi, cioè "ri-inizializzarlo", in alcuni punti del vostro programma. Se per svolgere questa operazione sono sufficienti le tre istruzioni che abbiamo già incontrato negli esempi precedenti, allora probabilmente sarà meglio riscriverle ogni volta. Ma se invece di sole tre istruzioni questa operazione dovesse richiederne molte di più, allora potreste pensare di rieseguire più volte lo stesso pacchetto di istruzioni facendo sì che le variabili coinvolte assumano ogni volta il giusto valore. In maniera grafica le cose possono essere spiegate così:



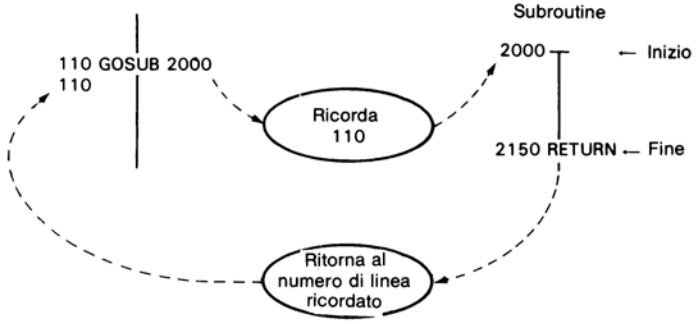
Se ora pensiamo di separare le istruzioni che ci interessano e di fare dei salti ad esse, **realizziamo quello che in gergo vien detta una "subroutine" (o "sottoprogramma").** Questo modo di procedere può non sembrare una novità perchè basta ogni volta porre una istruzione di GOTO alla linea ove inizia la subroutine. Ma come fa la

subroutine a ritornare a punti diversi del programma principale se come abbiamo postulato le sue istruzioni sono sempre le stesse?



Istruzione GOSUB

Abbiamo visto che il problema concernente una subroutine è quello del ritorno al programma principale. Infatti se in un programma abbiamo due GOTO, per saltare ad una subroutine, cioè vuole dire che vi saranno due diversi punti a cui possiamo ritornare quando la subroutine sia stata eseguita. **Per risolvere questo problema possiamo usare l'istruzione GOSUB che effettua il salto come la GOTO, ma in più ricorda dove si trova la successiva istruzione a cui ritornare dopo l'esecuzione della subroutine.** Questo può essere illustrato così:



Al fine di avere il corretto ritorno al programma principale è necessario che ogni **subroutine termini con l'istruzione RETURN**. RETURN fa appunto ritornare alla linea che era ricordata da GOSUB. Vediamo in un esempio come le tre linee per inizializzare il vettore A(I) possano diventare una subroutine:

```
10 REM  PROGRAMMA PRINCIPALE
20 REM POTETE DIMENSIONARE LE VARIABILI DELLE
30 REM SUBROUTINE NEL PROGRAMMA PRINCIPALE
40 REM E' PREFERIBILE PERO' DIMENSIONARE SEMPRE
50 REM TUTTE LE VARIABILI ALL'INIZIO DEL PRO-
55 REM GRAMMA PRINCIPALE
60 DIM A(99)
70 GOSUB 2000
80 REM VISUALIZZA QUALCOSA PER CONFERMARE
85 REM IL RITORNO DALLA SUBROUTINE
90 ?"OK RITORNO"
100 GOTO 100
2000 REM   SUBROUTINE!!
2010 FOR I=0 TO 99
2020 A(I)=I
2030 ?A(I);
2040 NEXT I
2050 RETURN
```

Subroutine nidificate

Le subroutine possono essere nidificate. Questo significa che una subroutine può chiamare un'altra subroutine che a sua volta può chiamarne una terza e così via. Per nidificare varie subroutine non dovete fare niente di difficile. Basta porre le istruzioni di salto GOSUB e terminare la subroutine con RETURN. Il BASIC CBM provvede da solo a ricordare con ordine i punti di ritorno. Ecco un esempio:

```
10 REM  PROGRAMMA PRINCIPALE
60 DIM A(99)
70 GOSUB 2000
80 REM VISUALIZZA QUALCOSA PER CONFERMARE
85 REM IL RITORNO DALLA SUBROUTINE
90 ?"OK RITORNO"
100 GOTO 100
2000 REM SUBROUTINE ESTERNA
2010 FOR I=1 TO 99
2020 A(I)=I
2030 GOSUB 3000
2040 NEXT I
2050 RETURN
3000 REM SUBROUTINE INTERNA
3010 ?A(I)
3020 RETURN
```

Questo programma ha spostato l'istruzione ? A(I) dalla prima subroutine ad un'altra.

Istruzione GOSUB calcolato

Le istruzioni GOTO e GOSUB sono molto simili. L'unica differenza è che GOSUB ricorda il punto di ritorno. Non dovete quindi meravigliarvi se esiste anche una istruzione GOSUB calcolato. Questa istruzione permette di saltare infatti ad

una tra diverse subroutine a seconda del valore assunto da un indice. Per esempio:

```
90
100 ON A GOSUB 1000,500,5000,2300
110
```

Quando il programma arriva alla linea 100 e se A = 1 viene richiamata la subroutine che inizia alla linea 1000. Se A = 2 viene invece richiamata la subroutine della linea 500. E così via A = 3 corrisponde alla subroutine della linea 5000 e A = 4 a quella della linea 2300. Se A assume un qualunque altro valore diverso da 1, 2, 3 o 4 il calcolatore vi darà un messaggio di errore e fermerà l'esecuzione del programma. Come abbiamo già detto l'istruzione GOSUB calcolato "ricorda" l'istruzione successiva prima di saltare alla subroutine (nell'esempio la linea 110).

Più istruzioni GOSUB possono essere nidificate proprio come le analoghe istruzioni GOTO.

Istruzione IF ... THEN

Nell'istruzione IF ... THEN si usano spesso gli operatori aritmetici e relazionali che abbiamo già descritto in questo capitolo. **Grazie a questo tipo di istruzione è possibile avere nei programmi delle capacità decisionali.** Infatti dopo IF dovete porre una espressione e dopo THEN una o più istruzioni. **Se l'espressione risulta "vera" allora le istruzioni dopo THEN saranno eseguite. Se invece risulta "falsa" le istruzioni dopo THEN non saranno eseguite e il programma passerà subito alla linea successiva.** Ecco un esempio:

```
10 IF A=B+5 THEN PRINT MSG1
40 IF CC$<"M" THEN IN=0
50 IF Q<14 AND M<>M1 GOTO 66
```

La parola THEN è opzionale e può essere omessa come nell'esempio della linea 50.

Alla linea 10, se A = B + 5 verrà stampato MSG1. Diversamente il messaggio non sarà stampato.

Alla linea 40 la variabile IN sarà posta eguale a 0 se la stringa CC$ rappresenta una lettera dell'alfabeto tra A e L.

Alla linea 50 si devono verificare due condizioni: se Q è minore di 14 e se M è diverso da M1, il programma salta alla linea 66.

Se in questo momento non ricordate come si calcola una espressione posta dopo IF, allora vi consigliamo di rivedere la prima parte di questo capitolo.

ISTRUZIONI DI INGRESSO E USCITA

Molte volte abbiamo detto che per ottenere la visualizzazione di un dato dobbiamo dare il comando PRINT oppure, in forma contratta, il punto interrogativo ?.

Come questo comando ne esistono molti altri che controllano il flusso dei dati da e verso il calcolatore. **Questi comandi vengono chiamati di "ingresso/uscita" o con termine inglese di "input/output" oppure generalmente di "I/O". I comandi più semplici e più frequenti sono quelli che regolano il trasferimento dei dati dalla tastiera al calcolatore e dal calcolatore al display video;** di questi ne parleremo qui di seguito. Esistono poi altri comandi più complessi che controllano il trasferimento dei dati tra il calcolatore e le periferiche più importanti come le unità a disco o a cassette magnetiche, le stampanti, ecc... Di questi comandi c'è ne occuperemo nel capitolo 6.

Cominciamo a vedere il comando PRINT che abbiamo già incontrato.

Istruzione PRINT

Ricordiamo che l'istruzione PRINT può essere contratta nella forma (?).

Qualcuno può chiedersi perchè diamo il comando "print", che vuol dire stampare, per visualizzare i dati su un display video e non diciamo invece "display". La spiegazione sta nel fatto che i primi piccoli calcolatori avevano come principale unità di uscita una stampante e non un terminale video perchè allora erano molto costosi.

L'istruzione PRINT visualizza un testo di stringa o dati numerici. La stringa deve essere racchiusa tra virgolette. Per esempio la seguente istruzione visualizza la parola "testo":

```
10 PRINT "TESTO"
```

Per visualizzare un numero potete porre dopo PRINT un numero o una variabile numerica:

```
10 A%=10
20 ?5,A%
```

L'istruzione alla linea 20 visualizzerà i numeri 5 e 10.

Potete mescolare numeri e stringhe, ma per separare le varie grandezze dovete porre delle virgole. In questo esempio l'istruzione PRINT visualizza le parole "uno", "due", "tre", ecc. e le corrispondenti cifre numeriche:

```
10 ?"UNO",1,"DUE",2,"TRE",3,"QUATTRO",4,"CINQUE",5
```

Se separate le variabili con delle virgole il calcolatore assegna automaticamente 10 caratteri di spazio ad ogni variabile.

Provate infatti a eseguire in modo immediato l'istruzione 10. Potete usare invece della virgola, per separare, il punto e virgola e allora non saranno lasciati spazi vuoti tra una variabile e l'altra. Per esempio:

```
10 PRINT "UNO";1;"DUE";2;"TRE";3;"QUATTRO";4;"CINQUE";5
```

Provate, come avete fatto prima, ad eseguirla in modo immediato.

L'istruzione PRINT esegue automaticamente un ritorno del carrello al termine della visualizzazione. Se però dopo l'ultima variabile ponete una virgola o un punto e virgola allora "il carrello non ritorna a capo". Se avete posto una virgola e se c'è un'altra istruzione PRINT nel vostro programma, allora la prima variabile di questa seconda PRINT verrà scritta dopo una spaziatura di 10 caratteri. Per vedere questo battete il seguente programmino e ponetelo in esecuzione con RUN:

```
10 PRINT "UNO",1,"DUE",2
20 PRINT "TRE",3,"QUATTRO",4
30 GOTO 30
```

A questo punto aggiungete una virgola in coda all'istruzione 10 ed eseguite nuovamente il programma con RUN. Vedrete che le due righe visualizzate diverranno una sola. È opportuno che vi ricordiate però che quando battete RUN lo dovete fare su una riga libera e non sopra al programma!

Provate ora a sostituire la virgola con il punto e virgola in coda all'istruzione 10 e poi eseguite il programma. Vi accorgerete che non vi sarà più spazio tra la cifra "2" e la parola "tre". Analogamente altri spazi tra variabili possono essere rimossi sostituendo la virgola con il punto e virgola nell'istruzione PRINT.

Nell'istruzione PRINT invece di inserire costanti numeriche è possibile porre nomi di variabili. Per esempio nella PRINT del programma precedente possiamo inserire la variabile A%(I) a cui assegneremo i valori interi da 1 a 5:

```
10 FOR I=1 TO 5
20 A%(I)=I
30 NEXT
40 PRINT "UNO";A%(1);"DUE";A%(2);"TRE";A%(3);"QUATTRO";A%(4);
   "CINQUE";A%(5)
50 GOTO 50
```

Quando è possibile, è più elegante porre l'istruzione PRINT in un ciclo FOR NEXT, come nell'esempio seguente in cui anche le stringhe sono state associate alla variabile N$(I):

```
10 DATA "UNO","DUE","TRE","QUATTRO","CINQUE"
20 FOR I=1 TO 5
30 A%(I)=I
40 READ N$(I)
50 PRINT N$(I);A%(I);
60 NEXT
70 GOTO 70
```

Questo programma può essere ulteriormente migliorato. Provate per esercizio a togliere la variabile A%(I) e non usare N$ come vettore, pur ottenendo alla fine la stessa visualizzazione sul display.

Funzioni speciali di stampa

Mediante opportuni comandi è possibile redigere, nella forma che ritenete più utile, la visualizzazione delle variabili e costanti di una istruzione PRINT. **Nel gergo dei calcolatori si dice che si stabilisce un "format" per un testo di stampa (o di visualizzazione).** Per esempio se volete scrivere un titolo in mezzo alla pagina potreste prevedere una stringa che contenga tutti i caratteri "spazio" necessari per portare il titolo in mezzo. Ma questo sarebbe ovviamente molto costoso, in termini di memoria del calcolatore, perchè dovreste memorizzare tutti questi spazi e per nulla elegante come programmazione. Per risolvere questi problemi di "format" sono state introdotte alcune opportune funzioni mediante le quali è possibile "arrangiare" come si vuole un testo da stampare (o visualizzare). Esse sono: SPC, TAB e POS.

Funzione SPC

Questa funzione da luogo ad una spaziatura di tanti caratteri quanto indicato dal numero posto come suo argomento SPC(N). Attenzione però che per spaziare solo di N caratteri, dopo SPC(N) dovete porre punto e virgola. Se ponete invece la virgola otterrete la stampa entro la decina di caratteri successiva. Per esempio per avere la parola inglese HEADING scritta in mezzo allo schermo da 40 colonne dovete dare l'istruzione:

```
10 ?SPC(16);"HEADING"
```

Per semplicità potete ricordare che la funzione SPC non fa altro che far spostare il carrello di tanti posti quanti sono indicati nel suo argomento. Tutte le altre regole dell'istruzione PRINT rimangono invariate.

Funzione TAB

TAB è l'equivalente della funzione tabulazione di una normale macchina da scrivere.

Supponete di dover stampare dei dati incolonnati. Sarà necessario allora stabilire da quali posizioni iniziano le varie colonne; per esempio:

NUMERO DI COLONNA
↓

0	16	32	48
JONES, P. J	431-25-6277	1420.00	258.74
BURKE, P. L	447-71-7614	2025.00	467.64
ROBINSON, L. W	231-80-8421	2150.00	477.04
etc.	etc.	etc.	etc.

In questo caso le colonne iniziano dalla posizione 0, 16, 32 e 48. (Ovviamente l'esempio si riferisce ad un display a 80 colonne).

Il modo di procedere più semplice, ma meno elegante, potrebbe essere quello di inserire gli spazi direttamente in una unica stringa che rappresenti tutta la riga:

```
10 ?"JONES,P.J              431-25-6277          1420.00          258.74"
```

Un'altra soluzione potrebbe essere data dall'uso della funzione SPC:

```
10 ?"JONES, P.J";SPC(17);"431-25-6277";SPC(5);"1420.00";SPC(9);"258.74"
```

Se procedessimo così riga per riga dovremmo ogni volta calcolare gli spazi in funzione delle parole presenti. **Con la funzione TAB viene invece stabilito in maniera univoca da quale posizione inizia ogni singolo dato.** Nel nostro esempio scriviamo infatti:

```
10 ?"JONES,P.J";TAB(16);"431-25-6277";TAB(32);"1420.00";TAB(48);"258.74"
```

Notate che i dati della terza e quarta colonna sono degli importi numerici che nell'esempio vengono stampati come stringhe. Se fossero stati trattati come numeri essi sarebbero stati allineati a destra e cioè in maniera diversa da quanto avviene per le stringhe. Provate da soli a riscrivere l'istruzione 10 usando variabili numeriche per i numeri e non variabili di stringa (se lo ritenete opportuno consultare il capitolo 5). Ricordate poi che per rappresentare i numeri viene lasciato uno spazio per l'eventuale segno meno e non vengono scritti gli zeri a destra nella parte decimale.

Funzione POS

La funzione POS è l'ultima delle funzioni di formato della PRINT. **POS ritorna la posizione in quel momento occupata dal cursore.**

Cioè comunica all'operatore un numero eguale alla posizione in cui si trova in quel momento il cursore. La funzione POS richiede che si ponga l'argomento 0.

Ecco un esempio:

```
10 ?"LA POSIZIONE DEL CURSORE E'";POS(0)
```

Otterrete come risposta dal vostro calcolatore:

```
?"LA POSIZIONE DEL CURSORE E' ";POS(0)
LA POSIZIONE DEL CURSORE E' 27
```

Se cambiate la lunghezza della stringa posta nella PRINT, vedrete cambiare il valore ritornato dalla funzione POS.

Istruzione INPUT

Mediante l'istruzione INPUT il calcolatore si pone in attesa di ricevere dati dalla tastiera. Sino a che tutti i dati richiesti non sono stati caricati attraverso la tastiera, nessuna altra operazione può essere eseguita.

Quando scrivete l'istruzione INPUT dovete farla eseguire da una lista di una o più variabili. Quando poi l'istruzione sarà eseguita ad ogni variabile verrà assegnato un valore caricato attraverso la tastiera. È ovvio che a seconda del tipo di variabile, dovrà essere assegnato un valore nel corrispondente formato. Per esempio se una variabile è il nome di una stringa (cioè termina con $) allora dovremo battere alla tastiera un testo la cui lunghezza in caratteri può essere qualunque. Per meglio capire quanto detto provate ad eseguire questo piccolo programma:

```
10 INPUT A$
20 ?A$
30 GOTO 10
```

Il calcolatore, per avvisarvi che sta eseguendo una istruzione INPUT e quindi aspetta da voi dei dati, vi stampa un punto interrogativo ? a cui dovete far seguire il dato numerico o stringa (nell'esempio è una stringa). In questo programmino abbiamo inserito anche una istruzione PRINT (linea 20) per farvi dare la conferma di quanto avete caricato.

Come abbiamo già detto se le variabili nella lista della INPUT sono numeriche allora dopo il punto interrogativo, che vi vien dato come segnale di avviso, dovete battere dei valori numerici. Se nella lista della INPUT compaiono più variabili esse devono essere separate da virgole. Tali virgole in questo caso non hanno altro significato che quello di separatori. Vi mostriamo un esempio per l'ingresso di una stringa, un numero con virgola e un numero intero:

```
10 INPUT A$,A,A%
20 ?A$,A,A%
30 GOTO 10
```

Dovete battere un testo, una virgola, un numero con virgola, un'altra virgola, un numero intero e infine un ritorno carrello. Se commettete un errore di formato il calcolatore vi avvisa con due punti interrogativi. Dovete ribattere allora quest'ultimo valore, ma se vi compare sul display un altro punto interrogativo seguito dal messaggio "re-do from start" ("ripetere dall'inizio") dovete ribattere tutti i dati.

Provate a cambiare l'ordine di A$, A e A% nell'istruzione PRINT e poi eseguite il programma.

Alcune volte è possibile assegnare un valore di tipo diverso, ma solo apparentemente, rispetto alle variabili listate nella INPUT. Infatti potete dare un valore intero, in corrispondenza ad una variabile numerica con virgola, in quanto esso sarà automaticamente convertito in forma decimale. Non potete però fare viceversa! Non potete far entrare una stringa con variabili numeriche, ma potete benissimo dare valori numerici ad una variabile di stringa. Provate da voi a fare qualche esempio con l'istruzione INPUT.

INPUT è una istruzione molto "esigente" e spesso un normale operatore non riesce a rispettare completamente la sua sintassi. Se i dati in ingresso devono essere dati da una persona non esperta di BASIC, dovrete scrivere dei programmi in grado

di prevenire ogni possibile errore di caricamento. Questi programmi, capaci di ovviare ad errori anche banali, vengono chiamati in gergo "a prova d'idiota". Nel capitolo 5 tratteremo in dettaglio programmi per l'ingresso dei dati.

Uno degli accorgimenti più semplici e più frequenti, per facilitare l'ingresso dei dati, è quello di usare la possibilità, che la INPUT offre, di visualizzare una stringa posta nella sua lista. Così in questa stringa possiamo far apparire un messaggio all'operatore concernente il formato delle variabili d'ingresso.

Per esempio:

```
10 INPUT "DAMMI IL NUMERO 1"; N
20 IF N<>1 THEN GOTO 50
30 ?"OK"
40 GOTO 40
50 ?"HAI SBAGLIATO!"
60 GOTO 10
```

Istruzione GET

L'istruzione GET permette l'ingresso di un singolo carattere e non richiede di premere il tasto RETURN. Il carattere battuto può essere un qualunque carattere tra quelli riconosciuti dal vostro calcolatore CBM oppure una cifra tra 0 e 9. Il dato in ingresso sarà interpretato come un carattere se nella lista di GET compare il nome di una stringa. Come in questo esempio:

```
10 GET A$
20 ?A$
30 GOTO 10
```

Quando ponete in esecuzione questo programma tutto ciò che è sullo schermo scorre via. Se provate a battere un carattere anche questo scorre via. La spiegazione di ciò è dovuta al fatto che l'istruzione GET non attende l'ingresso di un dato, ma suppone che sia già presente sullo schermo. **Se necessario possiamo fare in modo che GET attenda uno specifico carattere mediante una istruzione IF:**

```
10 GET A$
20 IF A$<>"X" THEN GOTO 10
30 ?A$
40 GOTO 10
```

Questo programma attende che entri la lettera X e non svolge alcuna altra funzione.

L'istruzione GET può essere programmata così da poter ricevere ogni carattere della tastiera. Per spiegare questo dobbiamo prima ricordare che alla variabile di stringa, elencata nella GET, viene inizialmente assegnato il valore nullo 00 che non può essere ricevuto dalla tastiera, ma può essere imposto dal programma mediante due virgolette consecutive "". Ecco come può essere scritto il programma:

```
10 GET A$
20 IF A$="" THEN GOTO 10
30 ?A$
40 GOTO 10
```

Se l'istruzione GET riporta una variabile numerica, intera o con virgola, allora il dato in ingresso sarà interpretato come valore numerico. Inizialmente a tale variabile il calcolatore assegna il valore zero 0, che può essere però dato dalla tastiera, per cui vi può essere l'indecisione se tale valore 0 è stato dato in ingresso oppure no. **Per questo motivo non si usa la GET per l'acquisizione di valori numerici.**

L'uso più comune della GET avviene in programmi di dialogo con l'operatore. Per esempio un programma può prevedere l'attesa di un operatore che quando sia presente debba battere una S per "si". Ecco come potremmo scrivere questa routine:

```
10 PRINT "CHIAMATA PER L'OPERATORE. BATTERE S PER SI"
20 GET A$
30 IF A$="S" THEN GOTO20
40 PRINT "OK"
```

Questo programma non riscrive il dato in ingresso. Provate a modificarlo così da fargli riscrivere quello che battete alla tastiera a seguito dell'istruzione GET.

ISTRUZIONI PEEK E POKE

Le istruzioni PEEK e POKE dovrebbero essere trattate nel capitolo 7, ma preferiamo descriverle qui perchè abbiamo dovuto già usarle.

I calcolatori CBM possono avere sino a 65536 posizioni di memoria indirizzabili, ognuna contenente un numero compreso tra 0 e 255. (Questo estremo superiore è $2^8 - 1$). **Tutti i programmi e i dati sono sempre convertiti in una sequenza di numeri memorizzati appunto nella memoria del calcolatore.**

L'istruzione PEEK vi permette di leggere il contenuto di qualunque posizione di memoria. Per esempio:

```
10 A%=PEEK(200)
```

Questa istruzione assegna il contenuto della posizione di memoria 200 alla variabile A%. L'argomento della PEEK può essere anche una variabile intera o una espressione intera, ma verrà sempre calcolato come valore intero per indicare una posizione di memoria.

L'istruzione POKE scrive invece un valore in una posizione di memoria. Per esempio:

```
20 POKE 8000,A%
```

scrive il contenuto della variabile A% nella posizione 8000. L'argomento della POKE può essere un numero, una variabile o una espressione con valore tra 0 e 255. Un valore con virgola viene convertito in intero.

Nelle memorie a sola lettura ROM potete applicare solo l'istruzione PEEK, mentre in quelle RAM a lettura/scrittura è evidente che potete applicare sia la PEEK che la POKE.

ISTRUZIONI END E STOP

Le istruzioni END e STOP fermano l'esecuzione del programma. Se però lo ritenete necesario il programma può continuare battendo CONT sulla tastiera. Non è obbligatorio inserire queste istruzioni in un programma, ma ne rendono la stesura più corretta.

In molti degli esempi precedenti, per fermare l'esecuzione del programma, abbiamo posto una istruzione di salto a se stessa. Per esempio:

```
50 GOTO 50
```

continuerà indefinitivamente ad eseguire il salto dalla linea 50 alla stessa linea. Diversamente avremmo potuto porre l'istruzione STOP e avremmo avuto il seguente messaggio:

```
BREAK IN XXXX
READY
```

che ci conferma che il programma si è fermato alla linea XXXX. Questo avviso è molto importante, se avete più di uno STOP nel vostro programma, perchè vi viene detto dove il programma si è fermato.

FUNZIONI

Le funzioni sono un altra componente molto importante del BASIC CBM. Esse sono simili sia alle variabili che alle istruzioni.

Forse il modo più semplice per capire che cosa è una funzione è fare un esempio:

```
10 A=SQR(B)
```

La variabile A è posta eguale alla radice quadrata di B. SQR è appunto la funzione radice quadrata ("Square root"). Ecco invece una funzione di stringa:

```
20 C$=LEFT$(D$,2)
```

La stringa C$ è posta eguale ai primi due caratteri della stringa D$.

Le funzioni possono sostituire qualunque variabile o costante di una istruzione BASIC tranne il caso di una variabile a sinistra del segno =. In altre parole potete scrivere A = SQR(B) ma non potete mai porre SQR(B) = A!

Abbiamo già incontrato alcune funzioni come SPC, TAB, POS e PEEK.

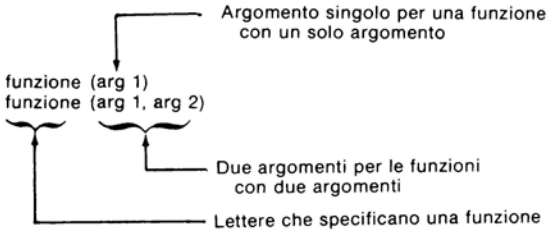
Vi illustriamo ora come usare le funzioni e vi diamo le specifiche di quelle principali. **Nel capitolo 8 potrete invece trovare la descrizione dettagliata di tutte le funzioni del BASIC CBM.**

Una funzione è caratterizzata da un nome, costituito da una o più lettere, e da uno o più argomenti posti tra parentesi. Per esempio SQR è il nome della funzione

"radice quadrata di" e B il suo argomento: SQR(B).

Nell'altro esempio C$ = LEFT$ (D$,2), LEFT$ è il nome della funzione mentre D$ e 2 sono gli argomenti.

In termini generali una funzione può avere uno di questi due formati:



Alcune funzioni richiedono però tre argomenti.

Importante: ogni argomento può essere una costante, una variabile o una espressione.

Una funzione che appaia in una istruzione sarà calcolata prima di ogni altro operatore. Cioè essa sarà ridotta ad un singolo valore numerico o ad una stringa prima di procedere al calcolo delle altre componenti dell'istruzione. Per esempio in questa istruzione:

```
10 B=24.7*(SQR(C)+5)-SIN(0.2+D)
```

SQR e SIN sono calcolate per prime. Supponiamo che SQR(C) = 6.72 e SIN(0.2 +D) = 0.625; la linea 10 diviene:

```
10 B=24.7*(6.72+5)-0.625
```

che non presenta alcun problema di esecuzione.

FUNZIONI ARITMETICHE

Le funzioni aritmetiche che potete usare nel BASIC CBM sono le seguenti:

INT	Converte l'argomento frazionario in un valore intero per troncamento
SGN	Ritorna il segno dell'argomento: + 1 per i numeri positivi, −1 per quelli negativi e 0 per l'argomento nullo
ABS	Ritorna il valore assoluto dell'argomento. I valori positivi restano inalterati, i negativi sono convertiti in positivo
SQR	Calcola la radice quadrata dell'argomento
EXP	Calcola la potenza e^{arg} dove "e" è la base dei logaritmi naturali

LOG	Calcola il logaritmo naturale dell'argomento
RND	Genera un numero a caso (vedere il capitolo 5)
SIN	Ritorna il valore trigonometrico seno dell'argomento; l'argomento è espresso in radianti
COS	Ritorna il valore trigonometrico coseno dell'argomento; l'argomento è espresso in radianti
TAN	Ritorna il valore tangente dell'argomento; l'argomento è espresso in radianti
ATN	Ritorna il valore arcotangente dell'argomento; l'argomento è espresso in radianti

Cercate di usare sino da ora le funzioni nei vostri programmi. Se però non comprendete il significato matematico di qualcuna, come forse SIN, COS e TAN, non preoccupatevene e concentrate la vostra attenzione sulle funzioni che conoscete bene.

Ecco un esempio di programma che usa la funzione INT:

```
10 A=2.743
20 B=INT(A)+7
30 ?B
40 STOP
```

Quando eseguite questo programma il risultato che viene visualizzato è 9 poichè la parte intera di A è 2. Per esercizio provate a cambiare la linea 10 con INPUT e la linea 40 con GOTO 10. Otterrete un programma ciclico che vi permette di analizzare come opera la funzione INT.

Il seguente esempio si presenta invece più complesso:

```
10 INPUT A,B
20 IF LOG(A)<0 THEN A=1/A
30 ?SQR(A)*EXP(B)
40 GOTO 10
```

Se conoscete già la funzione LOG provate allora a cambiare la linea 20 con un'altra funzione matematica.

L'argomento di una funzione può essere ancora una funzione.
Ecco un esempio:

```
30 ?SQR(A*EXP(B)+3)
```

Provate a usare funzioni aritmetiche in modo immediato inserendole, dopo una PRINT, in una espressione aritmetica.

FUNZIONI DI STRINGA

Le funzioni di stringa permettono di elaborare in vari modi le stringhe. Potete non conoscere alcune funzioni matematiche, ma dovete invece sapere usare molto bene tutte le funzioni di stringa.

Le funzioni di stringa del BASIC CBM sono le seguenti:

STR$	Converte un numero nel suo equivalente come stringa di caratteri
VAL	Converte una stringa numerica nel suo numero equivalente (solo se il numero assume un valore accettabile dal calcolatore)
CHR$	Decodifica un codice numerico ASCII
ASC	Codifica nel codice numerico ASCII
LEN	Ritorna il numero di caratteri di una stringa
LEFT$	Estrae la parte sinistra di una stringa
RIGHT$	Estrae la parte destra di una stringa
MID$	Estrae la parte interna di una stringa

In generale le funzioni di stringa vi permettono di determinare la lunghezza di una stringa, di estrarne una parte e di convertire i suoi caratteri in numeri con il codice ASCII e viceversa. Queste funzioni possono avere uno o due o tre argomenti. Ecco alcuni esempi:

```
STR$(14)
LEN("ABC")
LEN(A$+B$)
LEFT$(ST$,1)
```

FUNZIONI DEL SISTEMA

Per completare la trattazione delle funzioni vi descriviamo anche le funzioni del sistema. Queste funzioni saranno però usate solo quando avrete una buona esperienza di programmazione. L'unica funzione che potete usare facilmente, sin da adesso, è quella che vi da l'ora come se il calcolatore fosse un orologio. Essa serve per segnare eventuali elaborati che richiedano di sapere quando sono stati eseguiti.

Le funzioni del sistema sono:

PEEK	Legge il contenuto di un byte di memoria
TI$, T1	Ritorna l'ora del giorno data dall'orologio del calcolatore
FRE	Ritorna il numero di byte ancora disponibili di memoria RAM
SYS	Trasferisce il controllo ad un sottosistema
USR	Trasferisce il controllo ad un programma scritto in assembler dall'utente

FUNZIONI DEFINITE DALL'UTENTE

Oltre alle funzioni artimetiche standard del BASIC CBM potete usare funzioni che voi stessi avete definito.

Non potete però definire mai funzioni di stringa! Ecco un esempio di come si usa DEF FN per creare una nuova funzione:

```
10 DEFFNP(X)=100*X
20 INPUT A
30 ?A,FNP(A)
40 GOTO 20
```

DEF FN è l'istruzione per definire una nuova funzione. Essa deve essere seguita dal nome di una variabile numerica con virgola. Nell'esempio la variabile è P per cui la funzione diviene FNP. Se la variabile fosse stata AB allora la funzione sarebbe stata FNAB.

Nella istruzione DEF FN dopo il nome della variabile deve comparire un argomento, anche lui sotto forma di variabile, posto tra parentesi.
Questo nome della variabile argomento ha però la caratteristica di valere solo "localmente" nell'ambito dell'istruzione DEF FN. Ciò significa che contemporaneamente nel programma può esistere una variabile con lo stesso nome che assume un valore indipendente dall'argomento di FN. Nell'espressione che compare sul lato destro di DEF FN possono essere presenti, ma ciò non è obbligatorio, sia la variabile argomento sia altre variabili del programma.

Quando la funzione FN viene richiamata, con uno specifico valore per l'argomento, tale valore entra allora nell'espressione di definizione DEF FN, mentre le altre variabili mantengono il valore da esse posseduto in quel momento. L'espressione viene quindi calcolata e il risultato viene assegnato alla FN.

CAPITOLO 5

CARATTERISTICHE DEI CALCOLATORI CBM

In questo capitolo descriviamo le caratteristiche hardware dei calcolatori CBM e le tecniche di programmazione.

CARATTERISTICHE HARDWARE

BUFFER DELLA TASTIERA

Se premete due tasti contemporaneamente, o premete un secondo tasto prima che il primo sia visualizzato, allora una delle battute andrebbe persa se la vostra tastiera non fosse dotata di "buffer". **Il "buffer" delle tastiere CBM permette di "ricordare" le battute sino a che esse siano visualizzate sul display.**

Questo buffer permette di ricordare una o più battute, fino ad un massimo di dieci, mentre le precedenti sono in elaborazione. Senza questo buffer battute molto veloci andrebbero perse. Per esempio se la battuta 2° viene data prima che la battuta 1° sia elaborata, allora il calcolatore CBM memorizza la battuta 2°. A tempo debito la battuta 2° viene prelevata dal buffer ed elaborata. Questa caratteristica delle tastiere CBM è molto utile perchè vi permette di battere i caratteri molto velocemente.

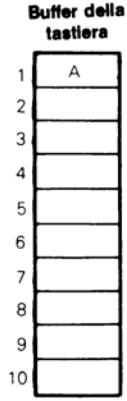
Descrizione del buffer

Tutti i calcolatori CBM hanno un "buffer" (memoria tampone) di 10 caratteri per memorizzare i caratteri che vengono battuti alla tastiera.

Per illustrarvi come funziona questo buffer caricate e ponete in esecuzione il programma PROVA della Figura 5-1. Premete quindi un tasto e mentre il display viene riempito con questo carattere battete dieci altri diversi caratteri sulla tastiera. Ogni carattere che avete battuto verrà prelevato e usato per riempire lo schermo.

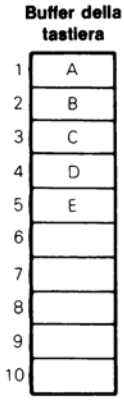
Vediamo più in dettaglio che cosa succede.

Ogni volta che premete un tasto il carattere corrispondente va ad ocupare la prima posizione del buffer della tastiera. Per esempio se premete A avrete:



Il calcolatore tiene conto sia del numero di caratteri presenti nel buffer sia della posizione del prossimo carattere che deve essere prelevato. Ogni volta che l'istruzione GET riceve un carattere un puntatore si sposta per indicare il prossimo carattere.

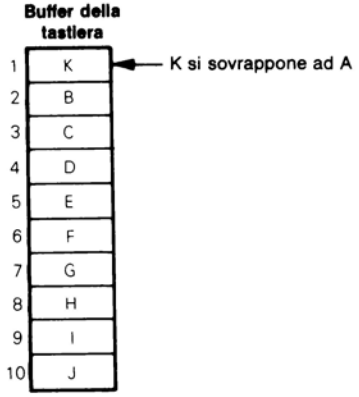
Se voi quindi battete dei caratteri, mentre il programma è occupato a visualizzare tutto lo schermo con A, allora questi caratteri andranno a occupare posizioni successive libere nel buffer. Nel disegno possiamo illustrare questo così:



Il programma PROVA, dopo aver visualizzato la A, preleverà il secondo carattere B e lo visualizzerà su tutto lo schermo. E così di seguito per tutti i caratteri che trova nel buffer.

Se avete battuto più di dieci caratteri allora il puntatore ritorna alla posizione 1

(per tutti i modelli CBM tranne che per la serie 8000). Per esempio se avete battuto le prime 11 lettere dell'alfabeto da A a K, allora la K andrà a sovrapporsi alla A:



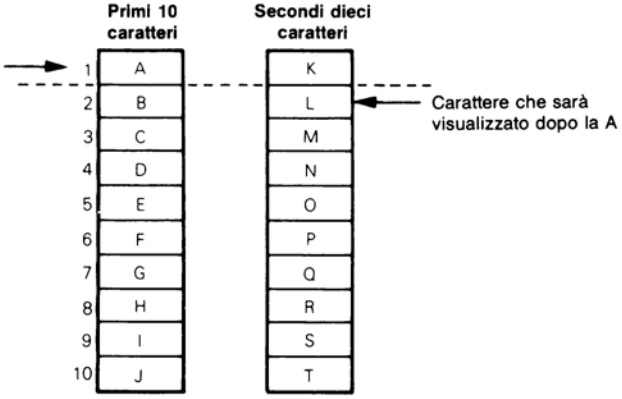
Succede così che quando il calcolatore va per prelevare un altro carattere ritrova il puntatore ancora nella prima posizione e "pensa" quindi che il buffer sia vuoto. Attenzione quindi che se battete 11 caratteri o multipli di 11 otterrete, nel program-

```
10 REM **************   PROVA   **************
20 REM
30 REM   VISUALIZZAZIONE CONTINUA DI UN
40 REM   CARATTERE BATTUTO ALL TASTIERA
50 REM
60 REM ****************************************
90 PRINT" BATTI UN TASTO O <R> PER TERMINARE"
100 GET C$: IF C$="" GOTO 100
105 IF C$=CHR$(13) GOTO 170
110 PRINT"□"          :REM PULISCE LO SCHERMO
120 FOR I=1 TO 920    :REM 920/40=23
130 PRINT C$;
140 NEXT
150 PRINT"UFFA!"
160 GOTO 90
170 END
```

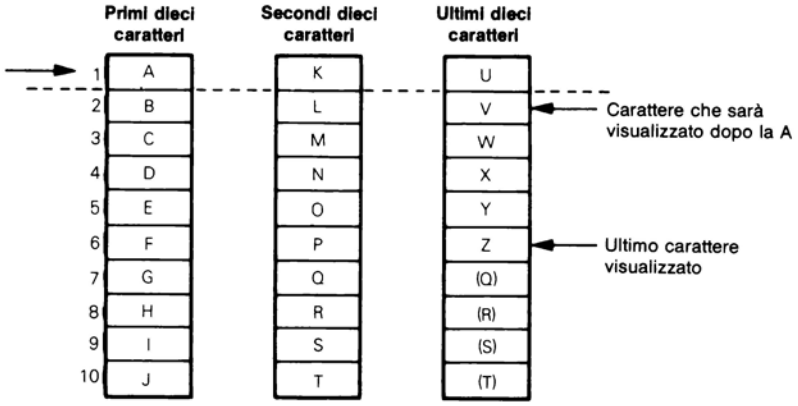
Figura 5-1: Programma PROVA

ma PROVA, lo stesso risultato come se ne aveste battuto uno solo.

Se batte invece 12 o 20 caratteri otterrete la stampa del primo e poi quella dal 12 in avanti. Per esempio battete A e poi durante la visualizzazione di A, battete B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S e T.



Otterrete sullo schermo le visualizzazioni di: A, L, M, N, O, P, Q, R, S e T.



L'effetto negativo si ottiene sempre ogni 11 caratteri. Se infatti battete inizialmente A e B e poi le lettere C, D, E, F, G, H, I, J, K e L otterrete che questi ultimi 10 caratteri siano cancellati via.

La serie CBM 8000 scarta i caratteri d'ingresso che gli altri modelli conservano con il buffer d'ingresso.

Svuotamento del Buffer prima dell'istruzione GET

Il buffer della tastiera può essere un valido aiuto, come nel caso del programma PROVA, ma può anche causare notevoli inconvenienti. **Se avete inavvertitamente battuto un carattere questo vi rimane nel buffer e può causare irregolarità nel programma. Per evitare questo potete programmare un ciclo per vuotare il buffer**

prima di leggervi una risposta attesa dalla tastiera. Questo ciclo può essere così scritto:

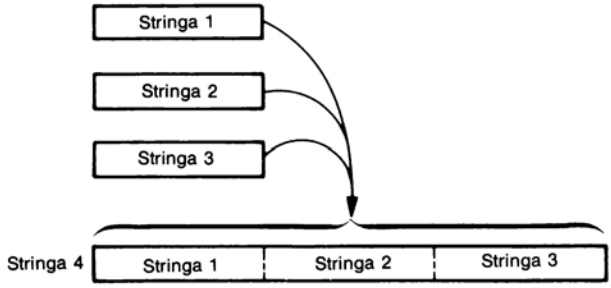
```
95 FOR I=1TO10 GET C$:NEXTI:
100 GET C$:IF C$="" GOTO 100
```

L'istruzione alla linea 95 vuota il buffer semplicemente leggendo tutti i suoi dieci valori.

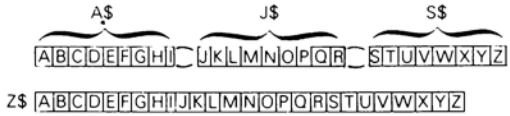
Se ora provate a eseguire il programma PROVA con l'aggiunta della linea 95 vi accorgerete che non potrete più caricare automaticamente il buffer con le lettere che volete visualizzare.

CONCATENAZIONE DI STRINGHE

Le stringhe CBM possono contenere sia caratteri alfabetici, che numerici, che grafici o una loro qualunque combinazione. Spesso è utile scrivere stringhe abbastanza corte e poi unirle tra loro come gli anelli di una catena.



Supponete per esempio che vogliamo creare una stringa Z$ che contenga tutte le lettere dell'alfabeto. Per fare questo possiamo unire tre stringhe che contengono ciascuna una parte dell'alfabeto. Il disegno che segue vi illustra chiaramente il concetto:



Per unire due stringhe si usa il segno "+". Il segno di somma infatti oltre a sommare valori numerici permette anche di concatenare due stringhe. In Tabella 5-1 sono riportati esempi esplicativi a tale riguardo.

Tabella 5-1: Operazioni Addizione (+)

Segno	Tipo	Esempio di istruzione	Operazione	Risultato
+	numeri	P = 2 + 3	2 + 3	P = 5
+	variabili numeriche	Q = T + S T = 12345 S = 11111	12345 +11111 23456	Q = 23456
+	stringhe alfabetiche	R$ = A$ + F$ A$ = ABCDE F$ = FGHIJ	ABCDE⌣FGHIJ	R$ = ABCDEFGHIJ
+	stringhe numeriche	Q$ = T$ + S$ T$ = 12345 S$ = 11111	12345⌣11111	Q$ = 1234511111

Attenzione però! Non è possiebile separare due stringhe con il segno di sottrazione "–" come si potrebbe pensare in analogia all'uso del segno "+" per unirle. Per esempio per creare la stringa con le lettere dell'alfabeto da J a Z non è possibile "sottrarre" a Z$ la stringa A$ cioè scrivere:

```
X$=Z$-A$ ←——— Errato
```

Provate da voi caricando le stringhe A$, J$ e S$ e dando l'istruzione X$ = Z$ – A$. Otterrete il messaggio di errore ?TYPE MISMATCH ERROR IN LINE 50:

```
10 A$="ABCDEFGHI"
20 J$="JKLMNOPQR"
30 S$="STUVWXYZ"
40 Z$=A$+J$+S$
50 X$=Z$-A$ ←——— Errato
60 PRINT X$

RUN

?TYPE MISMATCH ERROR IN LINE 50
```

L'unico operatore aritmetico valido per le stringhe è il segno di addizione "+". Gli altri operatori "–★/" non sono validi sebbene siano accettati gli operatori Booleani "< > =" usati per il confronto di due stringhe.

Se dovete estrarre'parte di una stringa potete usare le funzioni di stringa che abbiamo già visto: LEFT$, MID$ e RIGHT$.

Il problema che ci siamo posti prima può essere così risolto:

```
50 X$=RIGHT$(Z$,17)
   X$ = RIGHT$(ABCDEFGHIJKLMNOPQRSTUVWXYZ,17)
   X$ = JKLMNOPQRSTUVWXYZ
```

oppure unendo J$ e S$:

```
50 X$=J$+S$
   X$=JKLMNOPQR+STUVWXYZ
   X$=JKLMNOPQRSTUVWXYZ
```

Concatenamento su schermo o stampante

Se il concatenamento deve avvenire solo su un supporto di uscita come il display o una stampante, allora potete usare semplicemente l'istruzione PRINT con il segno di punto e virgola come separatore. Infatti se proviamo a scrivere:

```
PRINT A$;J$;S$

ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

otteniamo il concatenamento apparente delle stringhe.

STRINGHE GRAFICHE

Le stringhe di caratteri grafici possono essere concatenate come qualunque altra stringa. Questo permette di tracciare facilmente diagrammi e disegni.

STRINGHE NUMERICHE

Le stringhe numeriche sono caratterizzate dal fatto di contenere solo caratteri numerici. Esse possono essere create in due maniere che portano a risultati leggermente diversi.

Quando una variabile numerica viene assegnata ad una stringa numerica mediante la funzione STR$, anche il suo segno viene trasferito (uno spazio per i numeri positivi e il segno "–" per i negativi). Ecco un esempio:

```
10 AB=12345
20 T$=STR$(AB)
30 PRINT"AB=";AB
40 PRINT"T$=";T$

RUN

AB= 12345
T$= 12345
```

Se invece creiamo la stringa numerica ponendo direttamente tra le virgolette delle cifre, dobbiamo prevedere noi se il numero è positivo o negativo (cioè porre lo spazio oppure il segno meno).

Ecco un esempio:

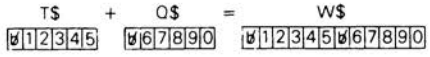
```
10 AB=12345
20 T$="12345"
30 PRINT"AB=";AB
40 PRINT"T$=";T$

RUN

AB= 12345 ←—— Uno spazio viene inserito
T$=12345 ←—— Nessuno spazio
```

Concateniamo ora due stringhe numeriche T$ e Q$ per creare la nuova stringa W$. Supponiamo per esempio che W$ debba contenere tutte le cifre 1, 2, 3, 4, 5, 6, 7, 8, 9 e 0. Allora possiamo scrivere:

```
10 T=12345
20 Q=67890
30 T$=STR$(T)
40 Q$=STR$(Q)
50 W$=T$+Q$ ←—— Crea una nuova stringa W$
60 PRINT"W$=";W$

RUN

W$= 12345 67890
```

Perchè compaiono due spazi davanti a 1 e 6? Questo è avvenuto perchè i due numeri originari T e Q erano positivi e quindi è stato mantenuto il loro segno.

Di conseguenza quando le due stringhe T$ e Q$ sono state concatenate, per formare W$, sono rimasti i due spazi che rappresentavano il segno più.



Vediamo adesso come possiamo fare per togliere questi spazi bianchi. Siccome noi volevamo conservare sole le cifre numeriche delle stringhe T$ e Q$ possiamo allora usare le funzioni LEFT$, MID$ e RIGHT$ per estrarre quella parte delle stringhe che ci interessa. Nel nostro caso noi vogliamo tutti i caratteri a destra del primo per cui possiamo scrivere T$ = RIGHT$ (T$, LEN (T$)−1):

In pratica diciamo al calcolatore di considerare le stringhe T$ e Q$ a partire dal secondo carattere. Dopodichè possiamo concatenare le due stringhe:

```
W$=RIGHT$(T$,LEN(T$)-1)+RIGHT$(Q$,LEN(Q$)-1)
```

Toglie gli spazi anteriori di T$ — Concatenamento di T$ con Q$ — Toglie gli spazi anteriori di Q$

In definitiva possiamo scrivere questo programma:

```
10 T=12345
   T = ø12345
20 Q=67890
   Q = ø67890
30 T$=STR$(T)
   T$ = ø12345
40 Q$=STR$(Q)
   Q$ = ø67890
50 W$=RIGHT$(T$,LEN(T$)-1)+RIGHT$(LEN(Q$)-1)
   W$ = RIGHT$(T$,6-1)          +RIGHT$(Q$,6-1)
   W$ = RIGHT$(T$,5)            +RIGHT$(Q$,5)
   W$ = T$  12345               +Q$  67890
   W$ = 1234567890
60 PRINT "W$=";W$
RUN
W$=1234567890
```

Avrete notato però che non è stato fatto alcun controllo dei segni dei due numeri. Se ambedue fossero negativi allora bisogna mantenere il segno contenuto in T$ che verrebbe trasferito a W$. Se infine le due stringhe avessero segni diversi non potrebbero essere concatenate.

PROGRAMMAZIONE DI INGRESSO E USCITA

Appena comincerete a programmare vi accorgerete che la parte di un programma che interessa le operazioni di ingresso e uscita è la più delicata.

Quasi tutti i programmi necessitano di dati che devono essere dati dalla tastiera e spesso questi dati sono molto numerosi. Ma le difficoltà non riguardano solamente la quantità dei dati bensì anche la possibilità dell'operatore di commettere errori.

Un programma efficiente deve prevedere che l'operatore possa umanamente commettere errori.

Analogamente nella fase di uscita non basta dare una sfilza di PRINT per stampare il risultato di un lavoro, ma bisogna anche rendere facilmente leggibile lo stampato di uscita.

Fortunatamente il BASIC CBM è ricco di istruzioni che permettono di programmare in forma molto professionale le operazioni d'ingresso e uscita. Analizzeremo adesso alcune di queste possibilità prima di entrare nei dettagli di una buona tecnica di programmazione.

ISTRUZIONE PRINT

Uso del "punto e virgola"

Normalmente l'istruzione PRINT comporta un ritorno del carrello al termine della sua visualizzazione. Questo significa che una seconda PRINT inizierebbe ad operare all'inizio della riga successiva. Osservate attentamente questa linea di programma; tutte le W sono visualizzate all'inizio di una nuova linea:

```
C$="W":FOR I=1 TO 800:? C$;:NEXT:?"UFFA!"
W
W
W
W
W
W
W
W
W
W
W
W
W
W
W
W
W
W
W
W
W
"UFFA!"

READY.
```

Un segno di "punto e virgola", che appaia dopo una variabile in una istruzione PRINT, darà luogo alla stampa della variabile seguente nella posizione subito successiva. Se quindi l'ultimo carattere di una istruzione PRINT è il ";" allora non avremo il ritorno a capo. In questo semplice programma le W sono appunto scritte tutte di seguito su 20 righe e 40 colonne.

```
C$="W":FOR I=1 TO 20:? C$:NEXT:?"UFFA!"
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
UFFA!

READY.
*
```

La variabile I nel ciclo FOR NEXT è usata come contatore per calcolare il numero di W da visualizzare. La prima W appare ovviamente all'inizio di una riga, ma poi tutte le altre sono poste di seguito tranne il caso in cui una riga dello schermo è piena e allora forzatamente si effettua un ritorno a capo.

Non pensiate che l'esclamazione UFFA sia stata stampata su una nuova riga. In realtà l'ultima W era stata stampata nell'ultima posizione della riga precedente per cui è avvenuto automaticamente un ritorno a capo. Se cambiate infatti l'estremo 800 con 780 vedrete che UFFA è visualizzata sempre dopo le W:

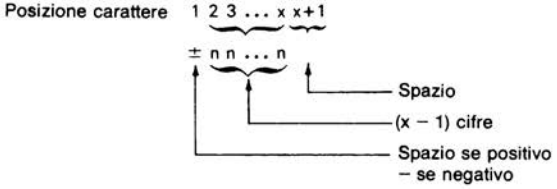
```
C$="-":FOR I=1 TO 780:?C$;:NEXT:?"UFFA!"
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
---------------------- "UFFA!"

READY.
*
```

Ripetiamo ancora che con il ";" non viene lasciato alcuno spazio tra due stringhe. Nel caso invece di variabili numeriche viene lasciato uno spazio per grandezze negative e due spazi per quelle positive.

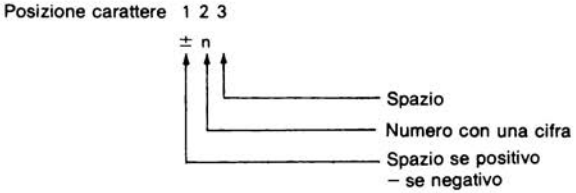
Per illustrare questo cambiate la stringa con il numero 5 nell'esempio e poi riducete a 267 l'estremo del ciclo in quanto ogni numero occupa tre spazi (800/3 = 267):

```
C=+5:FOR I=1 TO 267:?C;:NEXT:?"UFFA!"
 5  5  5  5  5  5  5  5  5  5  5  5  5
5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5
 5  5  5  5  5  5  5  5  5  5  5  5  5
5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5
 5  5  5  5  5  5  5  5  5  5  5  5  5
5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5
 5  5  5  5  5  5  5  5  5  5  5  5  5
5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5
 5  5  5  5  5  5  5  5  5  5  5  5  5
5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5
 5  5  5  5  5  5  5  5  5  5  5  5  5
5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5
 5  5  5  5  5  5  5  5  5  5  5  5  5
5  5  5  5  5  5  5  5  5  5  5  5  5  5
 UFFA!

READY.
```

Notate che tra l'ultimo 5 e la parola UFFA vi è un solo spazio. Questo è dovuto al fatto che i numeri sono visualizzati con il seguente formato:



che per una sola cifra diviene:



Se invece del numero 5 aveste posto un numero con più cifre allora avremmo avuto uno scorrimento verso l'alto del testo. Per esempio se poniamo il numero 2001 allora dobbiamo prevedere 6 spazi per ogni numero e per aggiustare le cose dobbiamo ridurre l'estremo del ciclo a 134 (800/6 = 134):

```
C=2001:FOR I=1 TO 134:?C;:NEXT:?"UFFA!"
 2001  2001  2001  2001  2001  2001  200
1  2001  2001  2001  2001  2001  2001  2
001  2001  2001  2001  2001  2001  2001
 2001  2001  2001  2001  2001  2001  200
1  2001  2001  2001  2001  2001  2001  2
001  2001  2001  2001  2001  2001  2001
 2001  2001  2001  2001  2001  2001  200
1  2001  2001  2001  2001  2001  2001  2
001  2001  2001  2001  2001  2001  2001
 2001  2001  2001  2001  2001  2001  200
1  2001  2001  2001  2001  2001  2001  2
001  2001  2001  2001  2001  2001  2001
 2001  2001  2001  2001  2001  2001  200
1  2001  2001  2001  2001  2001  2001  2
001  2001  2001  2001  2001  2001  2001
 2001  2001  2001  2001  2001  2001  200
1  2001  2001  2001  2001  2001  2001  2
001  2001  2001  2001  2001  2001  2001
 2001  2001  2001  2001  2001  2001  200
1  2001  2001  2001  2001  2001  2001  2
001 UFFA!

READY.
```

Notate anche che i numeri talvolta sono spezzati alla fine di una riga. Questo vi conferma il fatto che il separatore ";" genera una visualizzazione continua.

Uso della "virgola"

La presenza della virgola dopo le variabili di una PRINT equivale ad una tabulazione della riga con intervalli di 10 caratteri. Per uno schermo a 40 colonne questo può essere così illustrato:

```
1          11          21          31
```

Posizione più a sinistra = 1

Nel programma precedente cambiate il punto e virgola con la virgola. I numeri saranno visualizzati in 4 colonne. Per non far scorrere il testo riducete l'estremo superiore della variabile I a 80 (4 ★ 20). Notate poi che la prima posizione è sempre riservata al segno.

```
C=2001:FOR I=1 TO 80:? C,:NEXT:?"UFFA!"

 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
 2001          2001          2001          2001
"UFFA!"

READY.
```

L'uso della virgola nell'istuzione PRINT permette di ottenere dei tabulati molto interessanti.

Provate infatti a cambiare il valore 2001 in 44 e l'estremo del ciclo da 80 a 200. Premete RETURN e vedrete apparire il seguente testo particolarmente strano:

```
C=44:FOR I=1 TO 200:? C,:NEXT:?"UFFA!"
 440 442 44  440 442 44  440 442 44  440
 442 44  440 442 44  440 442 44  440 442
 44  440 442 44  440 442 44  440 442 44
 440 442 44  440 442 44  440 442 44  440
 442 44  440 442 44  440 442 44  440 442
 44  440 442 44  440 442 44  440 442 44
 440 442 44  440 442 44  440 442 44  440
 442 44  440 442 44  440 442 44  440 442
 44  440 442 44  440 442 44  440 442 44
 440 442 44  440 442 44  440 442 44  440
 442 44  440 442 44  440 442 44  440 442
 44  440 442 44  440 442 44  440 442 44
 440 442 44  440 442 44  440 442 44  440
 442 44  440 442 44  440 442 44  440 442
 44  440 442 44  440 442 44  440 442 44
 440 442 44  440 442 44  440 442 44  440
 442 44  440 442 44  440 442 44  440 442
UFFA!FFA!

READY.
```

Parte del vecchio testo non viene cancellato e si sovrappone al nuovo! **Il BASIC CBM usa infatti il comando di spostamento a destra del cursore, tra i campi di un testo, per cui non cancella ciò che era già presente sullo schermo.** Notate che rimane anche una parte della parola UFFA.

Tutto ciò può essere di grande aiuto quando si devono aggiungere dati su un tabulato già presente sullo schermo per cui vi consigliamo di non dimenticare mai questo aspetto dell'istruzione PRINT.

Se invece dovete rimuovere dallo schermo vecchie informazioni, allora potete inserire una istruzione per pulire lo schermo prima di effettuare una nuova visualizzazione. Per fare questo aggiungete l'istruzione PRINT CLEAR SCREEN nel modo seguente:

```
C=44:?"":FOR I=1 TO 200: ?C;: NEXT: ?"UFFA!"
```

Pulisce schermo (tasto CLR/HOME shiftato)

Quando premete RETURN vedrete lo schermo pulirsi e le cifre 44 iniziare ad apparire dalla *seconda* riga.

Per ottenere la visualizzazione delle cifre sin dalla *prima* riga, inserite un punto e virgola dopo l'istruzione PRINT CLEAR SCREEN:

```
C$="A":?"";:FOR I=1 TO 840: ?C$;: NEXT: ?"UFFA!"
```

In questo caso il programma può visualizzare 840 caratteri senza dar luogo a scorrimento del testo verso l'alto.

È ovvio che la virgola può essere usata anche con la stampa di stringhe. Per esempio caricate il seguente programma in modo immediato che visualizza 20 righe con ognuna 4 stringhe:

```
A$="UNO!":B$="DUE!":C$="TRE!":D$="QUATTRO!"
FOR I=1 TO 20: ?A$,B$,C$,D$: NEXT: ?"UFFA!"
```

MOVIMENTI DEL CURSORE

Nel capitolo 3 abbiamo discusso le capacità di editing offerte dai tasti di controllo del cursore: CLEAR SCREEN/HOME, CURSOR UP/DOWN, CURSOR LEFT/RIGHT, INSERT/DELETE e RETURN.

I comandi CLEAR SCREEN/HOME, CURSOR UP/DOWN, CURSOR/LEFT/ RIGHT e REVERSE possono essere posti all'interno di una stringa in una istruzione PRINT.

Invece i comandi INSERT/DELETE e RETURN non possono essere usati in una PRINT.

I comandi del cursore sono interpretati come caratteri di una stringa posta nella lista di una PRINT. Consideriamo questo esempio:

```
100 PRINT"*▮▮*"
```

Le seconde virgolette cambiano il modo da differito a immediato

Movimenti del cursore differiti

Le prime virgolette cambiano il modo da immediato a differito

Quando l'istruzione PRINT sarà eseguita, vedrete il cursore muoversi a destra grazie allo spostamento dell'asterisco:

```
RUN

*  *
```

Per far pratica nella programmazione dei movimenti del cursore, battete questo programmino:

```
10  PRINT"<CLEAR SCREEN>"
20  PRINT"<CURSOR↓>*<CURSOR↓>*<CURSOR↓>*<REVERSE><CURSOR↑>*
    <CURSOR↑>*<CURSOR↑>*"
30  PRINT"<CURSOR↓><CURSOR↓><CURSOR↓><CURSOR↓>"
```

Il programma apparirà così sul vostro schermo:

```
10 PRINT"▯";
20 PRINT"▮*▮*▮*▯*▯*▯*";
30 PRINT"▮▮▮▮"
40 END
```

e in esecuzione vedrete:

```
      ▯
*    ▯
  *  ▯
    *
```

Probabilmente non è quello che vi aspettavate. Infatti se ritenevate che la sequenza di caratteri:

```
20 PRINT "▮*▮*▮*          "
```

portasse a far apparire gli asterischi in colonna verticale:

```
*

*

*
```

oppure che l'altra sequenza:

```
20 PRINT "
```

portassesi tre asterischi invertiti sui precedenti:

allora sicuramente **vi siete dimenticati dello spostamento automatico del cursore a destra dopo ogni battuta.** I comandi del cursore programmati portano a spostamenti verso l'alto o verso il basso, ma l'asterisco sarà "battuto" sempre a destra a causa dell'avanzamento automatico del cursore stesso. Ogni volta che un carattere viene visualizzato il cursore si sposta a destra per evitare la sovrapposizione accidentale di due caratteri. In questo disegno vi mostriamo i movimenti del cursore avvenuti nell'esempio di sopra:

Per compensare questo spostamento a destra del cursore dovete dare un comando di spostamento a sinistra prima di quelli verso l'alto o verso il basso. Per esempio questo programma visualizza tre asterischi posti verticalmente e tre altri asterischi posti accanto, ma sempre verticali:

```
20 PRINT "<CURSOR↑>*<CURSOR←><CURSOR↓>*<CURSOR←><CURSOR↓>*
    <REVERSE>*<CURSOR←><CURSOR↑>*<CURSOR←><CURSOR↑>*";
```

esso appare così sullo schermo:

```
20 PRINT"
```

Se tentate di programmare i comandi INSERT/DELETE e RETURN otterrete dei risultati sorprendenti.

Il comando INSERT non è programmabile anche se sembra esserlo. In fase di stesura del programma INSERT appare come "◫", ma quando passate all'esecuzione non ottenete alcun risultato.

Il tasto DELETE opera solo in modo immediato. Se provate a programmarlo in una istruzione PRINT otterrete semplicemente di cancellare il carattere precedente tra quelli in cui è inserito. DELETE è programmabile dopo un inserimento, ma

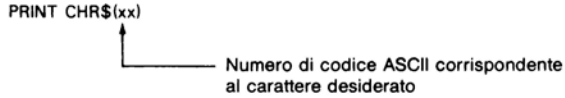
evitate di usarlo in questo modo perchè vi porta solo a delle complicazioni. Vi sono dei metodi più sicuri per ottenere lo stesso risultato.

Il comando RETURN in una PRINT muove immediatamente il cursore fuori dell'istruzione e lo porta alla linea successiva.

FUNZIONE CHR$ - PROGRAMMAZIONE DEI CARATTERI IN ASCII

Se non potete, o non volete, battere un tasto per inserire un carattere in una stringa, potete sempre ottenerlo tramite il suo valore in codice ASCII.

La funzione CHR$ converte un numero del codice ASCII nel suo carattere equivalente. Il formato di questa funzione è:



La tabella dei valori del codice ASCII è riportata nell'Appendice A. Cercate allora in questa tabella il valore che vi interessa e ponetelo come argomento della funzione CHR$. Per esempio per avere il carattere $ nella tabella del codice ASCII trovate 36 e 100. Ambedue vanno bene, ma una volta scelto uno è preferibile usare sempre lo stesso. Se per esempio prendiamo il valore 36, possiamo allora scrivere:

```
PRINT CHR$(36)
```

In modo immediato abbiamo:

```
PRINT CHR$(36)
$
```

Proviamo con l'altro valore 100:

```
PRINT CHR$(100)
$
```

Il risultato è lo stesso.

Provate a fare altri esempi di decodifica ASCII. La funzione CHR$ può essere inserita in una PRINT come se fosse una variabile:

```
10 PRINT CHR$(36);CHR$(42);CHR$(166)

RUN
$*▒
```

La funzione CHR$ vi permette di inserire tra i parametri di una PRINT quei caratteri che altrimenti non sarebbero disponibili come RETURN, INSERT/DELETE e le virgolette (“). Potete anche usarla in istruzioni di controllo, per esempio, nei

confronti dei comandi RETURN e INSERT/DELETE. Supponete che un programma debba ricevere dei caratteri dalla tastiera e sia in attesa del comando RETURN. Ricordando che l'equivalente ASCII di RETURN è 13 potete scrivere:

```
10 GET X$ IF X$<>CHR$(13) THEN 10
```

Questo test sarebbe impossibile se cercaste di porre RETURN tra virgolette:

```
20 IF X$<>"[RETURN]"THEN 10
```

Impossibile

Infatti appena premete RETURN il cursore si porta alla riga successiva.

```
20 IF X$<>"
```

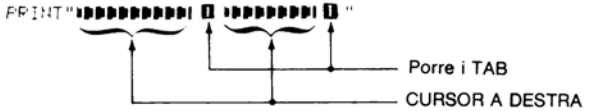
Premere tasto RETURN

Controlli del cursore (CBM 8000)

La versione 4.1 dell'editor per lo schermo, disponibile con i calcolatori CBM 8000, ha due nuovi tasti funzionali e alcune possibilità particolari di editing e controllo. Le due funzioni sono date dai tasti TAB e ESCAPE. Le possibilità di editing e controllo comprendono un campanello programmabile, l'inserimento e la cancellazione di linee, la cancellazione dello schermo, la commutazione tra caratteri grafici e di testo e lo scorrimento con una finestra programmabile sullo schermo.

Il tasto TAB e la funzione tabulazione. **Il tasto TAB opera in modo analogo a quello di una macchina da scrivere e le sue funzioni sono equivalenti a quelle della funzione TAB. Su una riga si possono predisporre sino a 80 punti di tabulazione.** Per usare TAB in modo immediato portate il cursore nella posizione voluta, quindi premete i tasti TAB e SHIFT contemporaneamente. Quando tutte le posizioni di tabulazione sono state poste premete RETURN:

Per tabulare a programma dovete porre TAB SET in una PRINT. Definite infatti una stringa che contenga tanti comandi CURSOR RIGHT quanti sono necessari per portare il cursore dove volete poi battete il tasto TAB shiftato che equivale a TAB SET. Questo può essere illustrato così:



Il carattere TAB SET è visualizzato con una I maiuscola in campo invertito.

Il comando TAB SET può essere anche dato mediante il suo valore in codice ASCII che è 137. Infatti con la funzione CHR$ potete scrivere:

```
PRINT"▮▮▮
```

Funzioni di controllo (CBM 8000)

Le funzioni di controllo, di cui qui di seguito ne riportiamo una breve descrizione, sono disponibili solo nei calcolatori CBM 8000 con lo schermo a 80 colonne. Queste funzioni saranno poi definite nei dettagli nel capitolo 8 e alcuni esempi saranno dati più avanti in questo capitolo.

Tutte queste funzioni sono previste per migliorare le visualizzazioni sullo schermo e l'ingresso dei dati, ma hanno la caratteristica di operare solo sullo schermo e di non modificare contemporaneamente il contenuto della memoria. Per questo motivo, anche se possono essere usate in modo immediato, non devono essere impiegate per l'editing di programmi.

Per usare queste funzioni dovete porre il loro carattere nella lista parametrica di una PRINT. Il carattere della funzione può essere indicato in una stringa mediante un carattere di controllo oppure mediante la funzione CHR$. Per generare un carattere di controllo premete il tasto ESCAPE, poi quello REVERSE e infine la corrispondente lettere senza shift.

Campanello ("Bell")

Questa funzione può operare solo con i calcolatori CBM 8000 che siano muniti di campanello. Il campanello suonerà sempre all'accensione del calcolatore e al superamento della colonna 75 dello schermo. Se la finestra dello schermo è stata ristretta (mediante le funzioni finestra dello schermo) il campanello suonerà appena il cursore passa la quinta posizione prima dell'estremo destro della finestra. Il campanello può essere anche fatto suonare mediante il comando Control-g oppure con CHR$(7) in una istruzione PRINT.

Inserimento e cancellazione di una linea ("Insert and delete line")

Queste funzioni inseriscono o cancellano una intera linea sullo schermo. "Delete line" cancella la linea su cui si trova il cursore e fa scorrere verso l'alto di un posto tutte le linee inferiori. Analogamente "insert line" crea una linea vuota facendo saltare verso il basso di un posto tutte le linee inferiori a quella dove c'è il cursore. La linea in fondo allo schermo esce dallo schermo. Ne l'inserimento di una linea ne la cancellazione modificano il contenuto della memoria del calcolatore. "Delete line" è generato da Control-u o da CHR$(21). "Insert line" è generato da Control-M o da CHR$(149).

Cancellazione dell'inizio o della fine di una linea ("Erase begin and erase end")

Queste funzioni cancellano parte della linea su si trova il cursore. "Erase begin" cancella tutti i caratteri alla sinistra del cursore mentre "erase end" cancella tutti quelli posti alla destra. Nessuna delle due funzioni modifica la memoria centrale.

"Erase begin" è generato da Control-V o da CHR$(150). "Erase end" è generato da Control-v o da CHR$(22).

Testo o caratteri grafici ("Text or graphic")

La funzione "graphic" seleziona i caratteri grafici dall'insieme di caratteri standard mentre i caratteri del testo possono essere lettere maiuscole o minuscole. Gli spazi tra i caratteri grafici sia sopra che sotto e a destra e sinistra, sono eliminati così da migliorare la qualità del disegno.
Questa funzione è generata da Control-N o da CHR$(142).

La funzione "text" è l'inverso della precedente e ne annulla gli effetti. I caratteri che hanno un simbolo grafico, nell'insieme di caratteri standard, sono tradotti nella rappresentazione dell'insieme di caratteri alternativo. La funzione "text" è generata da Control-n o da CHR$(14).

Funzioni finestra sullo schermo ("Screen window")

Vi sono quattro funzioni che permettono di definire una finestra sullo schermo del display e di far scrorrere in essa il testo. La funzione "set top" assume la posizione attuale del cursore come l'angolo di sinistra in alto della finestra. La funzione "set bottom" assume invece la posizione del cursore come l'angolo in basso a destra della finestra. Tale finestra può essere poi cancellata battendo il tasto HOME due volte di seguito oppure inserendo in una PRINT due caratteri HOME contigui. "set top" può essere selezionato con CHR$(15) mentre il "set bottom" con CHR$(143) posti in una PRINT dopo i caratteri di posizionamento del cursore.

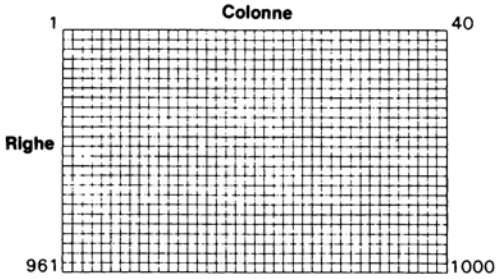
La funzione "scroll up" muove il testo all'interno della finestra di una linea verso l'alto e lascia vuota la linea più in basso della finestra. La funzione "scroll down" muove invece di una linea verso il basso il contenuto della finestra e lascia libera la linea più alta della finestra. "scroll up" è generato da Control-q o da CHR$(25). "Scroll down" è generato invece da Control-Q o da CHR$(153) posto in una PRINT come abbiamo sempre sottinteso.

Istruzione POKE applicata allo schermo

Mediante l'istruzione POKE potete visualizzare un carattere in qualunque punto dello schermo. **Per fare questo è sufficiente che diate il comando POKE per assegnare il valore del carattere nella corrispondente posizione di memoria.**

Lo schermo dei calcolatori CBM è come una griglia composta da 1000 (o 2000) quadratini ordinati su 25 righe e 40 colonne (o 80 colonne). Uno schermo a 40

colonne può essere disegnato così:

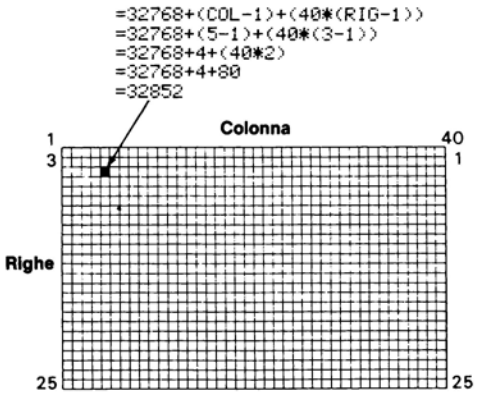


Qualunque carattere può essere visualizzato in ogni quadratino. **Ad ogni quadratino corrisponde una posizione della memoria centrale.** Per esempio al quadratino 1 (riga 1, colonna 1) corrisponde l'indirizzo 32768 e all'ultimo dello schermo a 40 colonne (riga 25, colonna 40) corrisponde l'indirizzo 33767. Nel caso invece dello schermo a 80 colonne all'ultimo quadratino (riga 25, colonna 80) corrisponde l'indirizzo 34767. In altre parole all'indirizzo 32768 corrisponde il quadratino (1,1), all'indirizzo 32769 corrisponde il quadratino (1,2), ecc. Nella Figura 5-2 trovate questa corrispondenza tra punti dello schermo e posizioni di memoria.

Per trovare facilmente l'indirizzo di memoria di un generico punto dello schermo vi diamo questa formula:

Schermo 40 colonne	**Schermo 80 colonne**
32768 + (colonna-1) + (40 · (riga-1))	32768 + (colonna-1) + (80 · (riga-1))

Provate a fare un esempio con il quadratino riga 3 e colonna 5:



Vedete appunto che l'indirizzo di memoria è 32852.

È allora possibile forzare un qualunque carattere sullo schermo mediante l'istruzione POKE conoscendo la riga e la colonna dove si vuole visualizzare il carattere.

Ricordiamo il formato della POKE:

```
POKE A,X
```

dove:

A è l'indirizzo sullo schermo
X è il carattere, o la variabile, da forzare in A

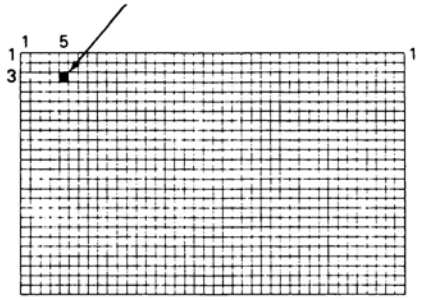
Possiamo sostituire direttamente al posto di A la formula per calcolare l'indirizzo di memoria:

$$\text{POKE } \underbrace{32768 + (COL - 1) + (40 \cdot ROW -)}_{A},X$$

Per esempio se COL(C) e RIG(R) sono dati in ingresso come 5 e 3 e per X diamo il simbolo delle picche ♠ allora avremo sullo schermo tale simbolo nella posizione (riga 3, colonna 5).

Provate infatti ad eseguire:

```
10 INPUT C,R,X
20 POKE 32768+(C-1)+(40*(R-1)),X

 =32768+(COL-1)+(40*(RIG-1)),X
 =32768+(5-1)+(40*(3-1)),X
 =32768+4+(40*2),X
 =32768+4+80,X
 =32852,X
```



X deve essere dato come valore in codice numerico ASCII, mentre sullo schermo apparirà il simbolo corrispondente.

In una istruzione POKE si possono usare delle variabili purchè esse assumano valori numerici nei limiti consentiti:

```
POKE 32768 + A,X
```

dove: A è un numero tra 0 e 999, per lo schermo a 40 colonne (32768 + 999 = 33767)

```
POKE A, X
```

dove:

A è un numero tra 32768 e 33767, per lo schermo a 40 colonne

Talvolta è molto comodo usare le variabili nell'istruzione POKE quando la si debba ripetere più volte come in questo caso:

```
10 A=32768
20 POKE A,X
30 A=A+10
40 IF A<=33767 GOTO 20
```

INGRESSO DATI (INPUT)

La fornitura di dati ad un calcolatore dovrebbe sempre essere programmata in termini di "unità funzionali".

Per meglio spiegare questo concetto facciamo un esempio. Un programma di gestione degli indirizzi richiede che in ingresso gli siano forniti nomi e indirizzi. Ogni nome e indirizzo costituisce appunto una "unità funzionale" cioè un tutt'uno che non dovrebbe essere mai separato. In altre parole il programma, quando chiede il nome e l'indirizzo, dovrebbe lasciare l'operatore libero di impostare come meglio crede la risposta. Solo quando egli ritenga di aver dato quella giusta, il programma deve acquisire il dato inteso appunto come "unità funzionale".

È una cattiva abitudine quella di acquisire ed elaborare i dati in piccole parti e trattarli come se fossero tra loro separati.

L'obiettivo dell'ingresso dati ("data entry") è quello di rendere il più facile possibile la fornitura dei dati e la correzione degli errori.

Supponete che un programma richieda in ingresso una lunga lista di dati molto brevi. Tale lista potrebbe essere costituita da nomi e date di nascita. Una buona idea è quella di accettare sul display un lungo blocco di questi dati. Così sin tanto che i nomi sono sul display possono essere corretti e verranno trasferiti al calcolatore solo quando saranno spinti in alto fuori dallo schermo ("scroll off").

Pensate come sarebbe invece rigida la soluzione elementare di trasferire subito al calcolatore ogni singolo nome e data di nascita.

Esistono casi di elaborazione in cui l'ingresso di dati in blocchi non è la soluzione migliore. Sorprendentemente questo si verifica quando bisogna caricare, tramite la tastiera, un numero molto elevato di dati. Per esperienza si è visto che i risultati migliori si ottengono se l'operatore carica, il maggior numero possibile di dati, senza badare agli errori durante una prima fase di caricamento. In altre parole in questa fase il calcolatore non deve correggere gli errori anche se sono stati rilevati. Il punto fondamentale di questo metodo di lavoro consiste nel far caricare una seconda volta gli stessi dati da un altro operatore. Successivamente poi il calcolatore confronterà i due ingressi e rigetterà quei dati che non siano eguali. Siccome la probabilità che ambedue gli operatori commettano gli stesse errori è molto piccola (trascurabile!) sarà facile individuare i dati errati perchè non sono eguali nei due caricamenti. Con un secondo programma si procederà poi alla loro correzione. Si è

potuto dimostrare che questo modo di caricare dati, anche se molto strano, porta a ottimi risultati.

Ingresso dati interattivo

Per mostrarvi come scrivere un buon programma per ingresso interattivo di dati, vi illustriamo un semplice esempio. Riprendiamo il programmino PROVA e passo passo vi facciamo vedere quali sono i miglioramenti da apportarvi.

Scriviamo dunque queste istruzioni:

```
100 C$="A"
110 PRINT "□"
120 FOR I=1 TO 840
130 PRINT C$;
140 NEXT
150 PRINT "UFFA!"
```

Questo programma visualizza 800 A seguite dall'esclamazione UFFA!

Supponiamo di voler cambiare le A con X. Per prima cosa cancellate l'istruzione di assegnazione 100. Per cancellare una istruzione vi ricordiamo che basta scriverne il numero e subito dopo battere RETURN.

```
LIST 100

 100 C$="A"

READY.
100 ←———————— Battere numero linea e RETURN
LIST

 110 PRINT "□";
 120 FOR I=1 TO 840
 130 PRINT C$;
 140 NEXT
 150 PRINT "UFFA!"
READY.
```

La linea 100 non è più nel programma. Battete ora C$ = "X" in modo immediato ed eseguite il programma.

Prima di RETURN	Dopo RETURN
C$="X"	"UFFA!"
READY.	READY.
RUN	

Lo schermo rimane bianco e alla fine compare UFFA! Ovviamente la X non è stata trasmessa al programma.

Il comando RUN azzera tutte le variabili numeriche e pone nulle tutte le stringhe

prima di eseguire il programma. Così la X è stata annullata e il programma ha visualizzato spazi vuoti.

Ci possiamo chiedere allora se esiste un modo per trasmettere al programma i valori C$ caricati in modo immediato. La risposta è positiva perchè possiamo dare il comando GOTO 110 che non cambia il valore di alcuna variabile.

Prima di RETURN

```
C$="X"

READY.
GOTO 110
```

Dopo RETURN

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
UFFA!

READY.
```

Allora la procedura per eseguire il programma è la seguente:

1. Caricate in modo immediato l'istruzione C$ = "y" dove y è un carattere qualunque.
2. Caricate in modo immediato l'istruzione GOTO 110.

Questo modo di procedere è però poco elegante; infatti vi mostriamo come sia possibile lavorare in maniera più semplice e professionale.

Il programma stesso può richiedere il valore del carattere da visualizzare con la seguente istruzione:

```
100 GET C$ IF C$="" GOTO 100
```

Quando ponete in esecuzione il programma vedrete che lo schermo si svuota e sparisce anche il cursore. Premete allora un qualunque carattere e lo vedrete

apparire 800 volte. Se rieseguite il programma e ribattete un altro carattere otterrete un'altra visualizzazione.

Questa seconda procedura è senz'altro migliore della prima, ma ha ancora il difetto di non avvisarvi che dovete dare un dato in ingresso. Aggiungete allora questa istruzione:

Copiare:

```
90 PRINT" BATTI UN TASTO"
```

Adesso il programma dà le istruzioni all'operatore. Provate ad eseguire il programma più volte e vedrete quanto è più facile usarlo.

C'è però ancora una modifica importante da fare. **Se volete eseguire il programma più volte potete terminarlo con una istruzione** di salto al suo inizio anzichè fermarlo. Aggiungete la seguente linea:

```
160 GOTO 90
```

Provate ora a listarlo:

```
LIST

 90 PRINT "BATTI UN TASTO"
 100 GET C$: IF C$="" GOTO 100
 110 PRINT "□"
 120 FOR I=1 TO 840
 130 PRINT C$;
 140 NEXT
 150 PRINT "UFFA!"
 160 GOTO 160
READY
```

ed a eseguirlo.

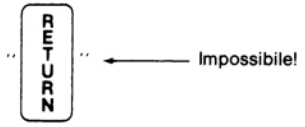
Ovviamente se adesso lo volete fermare dovete premere STOP. È più corretto però che poniate nel programma una interrogazione all'operatore per cui se premete un tasto, come per esempio RETURN, il programma si ferma.

Vediamo come fare.

Ogni tasto di dati o di controllo del cursore può essere interrogato come se fosse una stringa. Per esempio queste due linee interrogano il carattere Y:

```
100 GET C$ IF C$="" GOTO 100
105 IF C$="Y" GOTO 200
```

RETURN presenta invece dei problemi speciali. Non potete trattare RETURN come una stringa letterale, cioè:



Questo perche RETURN è un comando ad alta priorità per cui quando viene premuto il calcolatore memorizza subito quella linea e manda il cursore all'inizio della successiva.

Il problema può essere aggirato facendo uso della funzione CHR$ che converte i valori numerici del codice ASCII nei corrispondenti caratteri. Nel nostro caso il codice ASCII di RETURN è 13.

È importante però stabilire prima che cosa fare se verrà incontrato un RETURN. In tal caso ovviamente il programma si deve fermare per cui dobbiamo porre l'istruzione:

```
170 END
```

Poniamo allora:

```
105 IF C$=CHR$(13) GOTO 170
```

Oppure in modo più sintetico al posto della 105 e della 170:

```
105 IF C$=CHR$(13) THEN END ←— Opzione
```

In generale possiamo dirvi che, se ponete delle istruzioni per terminare un programma, è consigliabile che esse siano poste in coda perchè sarebbe molto scomodo andarle a cercare in mezzo al programma.

Nel nostro esempio non dovendo ricevere il messaggio READY per ogni esecuzione, rimangono libere due righe in più sullo schermo. È allora possibile visualizzare 80 caratteri in più per cui potete cambiare nella linea 120 l'estremo superiore del ciclo da 840 a 920:

```
120 FOR I=1 TO 920
```

Quando porrete in esecuzione il programma vedrete che una riga scorrerà in alto per cui ne rimarrà una vuota in basso. Questo è dovuto al fatto che il calcolatore esegue un RETURN dopo aver dato il messaggio BATTI UN TASTO per prepararvi per una nuova visualizzazione. Possiamo dimostrare questo rendendo il cursore visibile.

Normalmente durante l'esecuzione di un programma il cursore viene inibito. Mediante questa istruzione potete però renderlo visibile anche durante l'esecuzione di un programma:

```
30 POKE 548,0←——— Abilita il cursore
```

Per meglio comprendere questa istruzione vi rimandiamo al capitolo 7. Eseguite allora il programma e vedrete che il cursore è presente:

```
UFFA!
BATTI UN TASTO
```

L'istruzione 80 non è però molto importante per cui la potete cancellare.

Un altro metodo, per evitare la riga vuota in fondo allo schermo, è quello di porre un punto e virgola in coda alla PRINT della linea 90. In questa PRINT aggiungete anche un messaggio per avvisare che RETURN termina il programma. La linea 90 diviene così:

```
90 ?"BATTI UN TASTO O <R> PER TERMINARE";
```

Per terminare la nostra discussione sul programma PROVA vi consigliamo di porre alcuni commenti REM per spiegare i punti salienti del programma. Per esempio:

```
120 FOR I=1 TO 920          REM 920/40=23 LINEE
```

ed anche:

```
110 PRINT "Ω"               REM PULISCE LO SCHERMO
```

E da ultimo aggiungere quelche REM all'inizio di PROVA per spiegarne il contenuto. Otterrete a questo punto un programma come quello riportato in Figura 5-1. Registratelo allora su cassetta o su dischetto.

Messaggi di avviso

Ogni programma che preveda l'ingresso di dati deve avere dei messaggi di avviso da comunicare all'operatore. In genere sono messaggi molto semplici e brevi a cui l'operatore risponde SI o NO (YES o NO) oppure solamente Y e N. Un altro caso è quello in cui l'operatore risponde con qualche numero in codice.

Le parti di programmi, che sovrintendono a questi dialoghi è preferibile che siano poste sotto forma di subroutine per renderle indipendenti dalla filosofia del programma principale. Da questo ne derivano tre considerazioni:

1. Non potete presumere che la riga del display su cui apparirà il messaggio sia in quel momento libera. Di conseguenza il messaggio sarà sovrapposto ad un vecchio testo, e fin qui niente di male, ma il resto della riga sarà interpretato come risposta. Questa situazione può portare a delle grosse complicazioni.
2. Le subroutine devono ricevere dal programma principale alcuni parametri. Per esempio se il programma principale chiede dei valori numerici tramite la subroutine, allora le può passare gli estremi di validità di tali valori.
3. Le subroutine devono poi rimandare al programma principale le risposte dell'operatore che possono essere semplici caratteri come Y o N oppure parole o numeri.

Una subroutine non può sapere su quele riga dello schermo apparirà il messaggio di chiamata. È importante quindi stabilire che sia il programma principale a posizionare il cursore sulla riga giusta. La subroutine può poi pulire la riga e portare il cursore al suo inizio. Ecco le istruzioni che potete usare:

```
2000 REM PULISCE LA RIGA DEL CURSORE
2010 PRINT CHR$(13);"□";: REM MUOVE IL CURSORE SULLA COLONNA 0
2020 FOR I=1 TO 39: PRINT " ";: NEXT
2030 PRINT CHR$(13);"□";
2040 STOP
```

Nel caso di uno schermo a 80 colonne l'istruzione alla linea 2020 deve portare come estremo superiore il valore 79 invece di 39.

Caricate questo programma nel vostro calcolatore; posizionate il cursore su una linea bianca tra due linee di testo quindi battete RUN per eseguire il programma. Se tutto il testo scorre via dallo schermo allora vuol dire che avete dimenticato il punto e virgola in coda all'istruzione PRINT della linea 2020.

Spesso questo programma costituisce appunto una delle subroutine di cui abbiamo parlato per cui dovete porre RETURN alla linea 2040 invece di STOP.

Un altro modo di procedere prevede invece l'uso delle funzioni ERASE o ERASE END (solo per i calcolatori CBM 8000):

```
2000 REM PULISCE LA RIGA DEL CURSORE
2010 PRINT CHR$(150);CHR$(22);CHR$(13);"□";
2030 STOP
```

In tal caso la routine si riduce ad una unica linea di programma!

Vediamo ora una subroutine per porre una richiesta all'operatore a cui si deve rispondere con una Y (YES) o con una N (NO). Useremo una PRINT per porre la domanda e una GET per ricevere la risposta. Verrà inoltre pulita la riga su cui dare la risposta mediante la routine prima descritta. Ecco le subroutine:

```
2000 REM PULISCE LA RIGA DEL CURSORE
2010 PRINT CHR$(13);"□";: REM MUOVE IL CURSORE SULLA COLONNA 0
2020 FOR I=1 TO 39: PRINT " ";: NEXT
2030 PRINT CHR$(13);"□";
2040 RETURN
3000 REM PONE UNA DOMANDA E ATTENDE Y O N PER YN$
3010 GOSUB 2000
3020 PRINT"VUOI FARE DEI CAMBIAMENTI? ";
3030 GET YN$: IF YN$<>"N" AND YN$<>"Y" THEN 3030
3040 PRINT YN$;
3050 RETURN
```

Potete usare queste subroutine per chiedere qualunque cosa a cui si possa rispondere con un "sì" o un "no". Il messaggio per la richiesta deve essere posto nella PRINT della linea 3020.

Vediamo ora una subroutine che preveda come risposta un numero. Assumiamo che alla routine sia detto quale è il valore minimo accettabile LO% e quale è il

massimo HI%. La subroutine ritornerà il valore NM% dato in risposta. Ecco la subroutine:

```
2000 REM PULISCE LA RIGA DEL CURSORE
2010 PRINT CHR$(13);"□";: REM MUOVE IL CURSORE SULLA COLONNA 0
2020 FOR I=1 TO 39: PRINT " ";: NEXT
2030 PRINT CHR$(13);"□";
2040 RETURN
3000 REM CHIEDE UN VALORE NUMERICO
3001 REM CHE CONFERMA COME NM%.
3002 REM NM% DEVE ESSERE MINORE DI HI% E
3003 REM MAGGIORE DI LO%. IL PROGRAMMA CHIAMANTE
3004 REM DEVE DARE I VALORI HI% E LO%
3010 GOSUB 2000
3020 PRINT"QUALE VUOI CAMBIARE? ";
3030 GET NM$: IF NM$="" THEN 3030
3040 NM%=VAL(NM$)
3050 IF NM%<LO% OR NM%>HI% THEN 3030
3060 PRINT NM$;
3070 RETURN
```

Provate a scrivere un programma che prima definisca i due valori LO% e HI% e poi richiami la subroutine 3000.

La subroutine 3000 accetta come risposta un numero con una cifra sola, provate da voi a modificarla così da poter accettare numeri con due cifre. Se non vi riuscite aspettate di leggere più avanti questo capitolo.

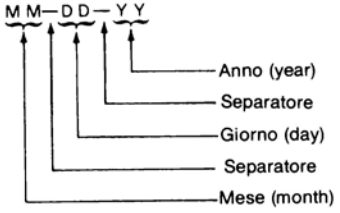
Se volete potete fare un'altra piccola modifica e cioè fornire alla PRINT, della linea 3020, il messaggio sotto forma di stringa che a sua volta può essere assegnata in maniera molto generale dal programma principale.

Ingresso e verifica di una data

Molti programmi durante la loro esecuzione chiedono all'operatore dei dati che non sono così semplici come un "sì" o un "no", ma neanche tanto complicati come un intero testo. Consideriamo per esempio la data.

Non sottovalutate questo esempio perchè contiene tutti gli elementi caratteristici di un buon "data entry".

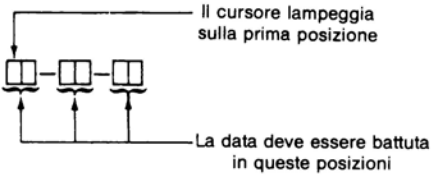
Poniamo che la data debba essere fornita così (con il mese prima del giorno come è d'uso nei paesi anglosassoni):



Se volete potete sostituire il trattino con una barra o un punto.

Programmate sempre l'ingresso dei dati in modo che esso sia facile e immediato

per l'operatore! Per esempio se dovete chiedere una data predisponete le posizioni dove l'operatore deve darla con un campo a caratteri invertiti; ciò sarà molto visivo ed efficacie:



Per creare il campo invertito potete usare questa PRINT:

```
10 PRINT "<Clear><Cursor↓><Cursor↓>";TAB(20);"<Reverse>␣␣
   <Reverse off>-<Reverse>␣␣<Reverse off>-<Reverse>
   ␣␣<Reverse off>";CHR$(13);"<Cursor↑>";TAB(20);

   ␣ rappresenta uno spazio
```

Questa istruzione PRINT comprende anche i comandi del cursore per posizionare l'ingresso dei dati alla colonna 21 della riga 3; e inoltre pulisce lo schermo per non avere vecchi caratteri nella zona che ci interessa. Dopo aver preparato il campo invertito il cursore viene riportato al suo inizio mediante un ritorno del carrello, un CURSOR UP e un TAB.

Provate ora a usare l'istruzione INPUT per ricevere il dato riguardante il mese. Per esempio così:

```
20 INPUT M$;
```

Caricate le istruzioni 10 e 20 ed eseguitele. L'istruzione INPUT non funzionerà. Innanzi tutto il punto interrogativo della INPUT occuperà il primo campo invertito della data e poi il resto della riga verrà preso come valore d'ingresso. Così sino a che non avrete riscritto su tutta la riga (ma questo richiede l'ingresso di dati molto lunghi) otterrete sempre il messaggio RE-DO FROM START ogni volta che premete RETURN.

È consigliabile quindi ritornare ad usare GET:

```
10 PRINT "▒▒";TAB(20);"▒ ▒-▒ ▒-▒ ▒";CHR$(13);"▒";TAB(20);
20 GET C$: IF C$="" THEN 20
30 PRINT C$;: MM$=C$
40 GET C$: IF C$="" THEN 40
50 PRINT C$;: MM$=MM$+C$
60 STOP
```

Queste istruzioni accettano due cifre in ingresso. L'ingresso è visualizzato nel primo campo invertito della data. Le due cifre date in ingresso non richiedono RETURN o altri caratteri terminali. Infatti il programma accetta solo i primi due caratteri che vengono battuti.

Per dare una data è necessario dare tre volte una coppia di numeri: una per il mese, una per il giorno e una per l'anno. Invece di ripetere tre volte le stesse istruzioni dalla linea 20 alla 50 le poniamo in una subroutine a cui facciamo tre volte riferimento:

```
10 PRINT "□□□";TAB(20);"■ ■-■ ■-■ ■";CHR$(13);"□";TAB(20);
20 GOSUB 1000: MM$=TC$: PRINT TAB(23)
30 GOSUB 1000: DD$=TC$: PRINT TAB(26)
40 GOSUB 1000: YY$=TC$
50 STOP
1000 REM SUBROUTINE INGRESSO DUE CARATTERI
1010 GET C$: IF C$="" THEN 1010
1020 PRINT C$;
1030 GET CC$: IF CC$="" THEN 1030
1040 PRINT CC$;
1050 TC$=C$+CC$
1060 RETURN
```

Se avete un calcolatore CBM 8000 provate a riscrivere questo programma facendo uso delle funzioni TAB SET e TAB fornite dalla versione 4.1 dell'Editor.

Sempre nel caso di un calcolatore CBM 8000 la versione di questo programma può essere molto più semplice perchè potete usare la funzione ERASE END come segue:

```
3000 REM ASK A QUESTION AND RETURN A RESPONSE OF Y OR N IN YN$
3005 REM CBM 8000 VERSION
3010 PRINT"□"CHR$(150);CHR$(22);CHR$(13);"□"
3020 PRINT"DO YOU WANT TO MAKE ANY CHANGES? ";
3030 GET YN$: IF YN$<>"N" AND YN$<>"Y" THEN 3030
3040 PRINTYN$;
3050 RETURN
```

Le variabili MM$, DD$ e YY$ contengono il mese, il giorno e l'anno espressi come stringhe di due caratteri. Ricordatevi che in questo caso dovete pulire il buffer, di dieci caratteri d'ingresso, prima di usare GET. Diversamente acquisireste qualunque vecchio carattere già contenuto nel buffer. Questa operazione è sufficiente che la facciate una sola volta.

Ci sono due modi per aiutare un operatore a correggere gli errori mentre sta caricando dati.

1. Il programma può controllare automaticamente la validità del mese, del giorno e dell'anno.
2. L'operatore può ripetere l'ingresso dei dati.

Il programma può controllare che il numero del mese sia compreso tra 1 e 12. Può controllare il numero massimo di giorni per ogni mese, eventualmente tener conto degli anni bisestili. Ogni anno tra 0 e 99 è valido. Se una di queste condizioni non è rispettata allora l'ingresso della data sarà ripetuto dall'inizio.

In definitiva il nostro programma per l'ingresso delle date appare così:

```
5 REM ROUTINE PER ACCETTARE E VERIFICARE UNA DATA
10 PRINT "[illegible]";TAB(20);"[illegible]";CHR$(13);"[illegible]";TAB(20);
50 GOSUB 1000:REM CHIEDE IL MESE
60 IFC$=CHR$(13) OR CC$=CHR$(13) THEN 10
70 DT$=TC$: PRINT TAB(23)
80 REM CONTROLLO VALIDITA' MESE
90 M%=VAL(TC$)
100 IF M%<1 OR M%>12 THEN 10
110 REM CONTROLLA LA LUNGHEZZA DEL MESE
120 D%=31
130 IF M%=2 THEN D%=28
140 IF M%=4 OR M%=6 OR M%=9 OR M%=11 THEN D%=30
150 GOSUB 1000: REM  CHIEDE IL GIORNO
160 IF C$=CHR$(13) OR CC$=CHR$(13) THEN 10
170 DT$=DT$+"-"+TC$: PRINT TAB(26)
190 REM CONTROLLO VALIDITA' GIORNO
200 IF VAL(TC$)<1 OR VAL(TC$)>D% THEN 10
210 GOSUB 1000: REM CHIEDE L'ANNO
220 DT$=DT$+"-"+TC$
230 IF C$=CHR$(13) OR CC$=CHR$(13) THEN 10
240 REM CONTROLLO VALIDITA' ANNO
260 IF VAL(TC$)<0 OR VAL(TC$)>99 THEN 10
270 STOP
280 REM
1000 REM SUBROUTINE PER INGRESSO DI DUE CARATTERI
1005 FOR I=1 TO 10: GET C$: NEXT: REM ANNULLA IL BUFFER D'INGRESSO
1010 GET C$: IF C$="" THEN 1010
1015 IF C$=CHR$(13) THEN 1050
1016 IF C$<"0" OR C$>"9" THEN 1010
1020 PRINT C$;
1030 GET CC$: IF CC$="" THEN 1030
1035 IF CC$=CHR$(13) THEN 1050
1036 IF CC$<"0" OR CC$>"9" THEN 1010
1040 PRINT CC$;
1050 TC$=C$+CC$
1060 RETURN
```

Notate che la data è posta nella stringa DT$ come mese, giorno e anno.

Sulla data fornita in ingresso vengono fatti questi tre controlli:

1. Il carattere è RETURN?
2. Se non è RETURN è una cifra valida?
3. La prima coppia di cifre costituisce un mese valido, la seconda un giorno valido e la terza un anno valido?

Il ritorno del carrello è stato scelto come carattere per far ripartire e ripristinare il programma. Se volete potete usare un qualunque altro carattere sostituendo CHR$(13) nelle linee 60, 160, 230 e 1035. Quando l'operatore preme RETURN o un altro carattere scelto per il ripristino, l'intero programma ricomincia da capo. Controlliamo il carattere di ripristino, alla linea 1035, poichè vogliamo che il ripristino avvenga dopo l'ingresso della prima o della seconda cifra. Anche il programma principale controlla la presenza del carattere di ripristino prima di tornare all'inizio della routine alla linea 10. Potremmo saltare questa fase e passare direttamente dalla subroutine alla linea 10, in caso di ripristino, senza ritornare prima regolarmente al programma principale. Questo modo di lavorare è però molto pericoloso e lo sconsigliamo assolutamente.

Ogni subroutine deve essere vista come una unità autonoma con i suoi punti fissi di ingresso e di uscita. Saltare da un punto qualunque di una subroutine al programma principale o da una subroutine ad un'altra è sicuramente motivo di gravi errori. In altre parole per uscire da una subroutine dovete sempre passare attraverso la sua istruzione RETURN.

Una eventuale routine, per controllare che in ingresso non siano dati caratteri non numerici, può risiedere nella stessa subroutine di ingresso dei due caratteri. Abbiamo infatti deciso di considerare errati caratteri non numerici dati in ingresso. Alle linee 1016 e 1036 controlliamo che i caratteri d'ingresso siano numerici facendo un confronto tra i loro valori ASCII e i valori ASCII delle cifre numeriche.

Il controllo di validità dei mesi, giorni e anni viene fatto nel programma principale.

L'istruzione alla linea 100 controlla la validità dei mesi. Le istruzioni alle linee 120, 130, 140 e 200 controllano la validità dei giorni tenendo conto della differente lunghezza dei mesi.

Alla linea 260 viene infine fatto il controllo dell'anno. Notate che alle linee da 90 a 140 abbiamo usato una rappresentazione numerica intera del mese per risparmiare memoria mentre non lo abbiamo fatto per i giorni e per gli anni.

Ricordatevi infine che se cercate di scrivere bene una routine di "data entry" voi forse perdete un poco di tempo, ma ne farete risparmiare molto di più a chi opererà con il vostro programma.

Moduli per ingresso dati

Se avete dati di ingresso complessi, il modo migliore per caricarli è quello di visualizzare sullo schermo un modulo e quindi riempirlo man mano con i dati. Considerate per semplicità il problema, già altre volte citato, di caricare nomi e indirizzi. Potreste allora visualizzare un modulo come questo:

```
BATTERE IL NOME E L'INDIRIZZO

 1    NOME:
 2     VIA:
 3  CITTA':
 4   STATO:            5  CAP:
```

Ad ogni dato in ingresso viene asegnato un numero che nel modulo è indicato in campo inverso. L'operatore non deve far altro che inserire i dati nei posti prescritti.

Il seguente programma pulisce lo schermo e visualizza il modulo:

```
10 REM PROGRAMMA PER L'INGRESSO DI
11 REM NOMI E INDIRIZZI
20 REM VISUALIZZA IL MODULO D'INGRESSO
30 PRINT"[CLR] BATTERE IL NOME E L'INDIRIZZO"
40 PRINT" [RVS]1[OFF]   NOME:"
50 PRINT" [RVS]2[OFF]    VIA:"
60 PRINT" [RVS]3[OFF] CITTA':"
70 PRINT" [RVS]4[OFF]  STATO:";TAB(28);"[RVS]5[OFF] CAP:"
```

Appena un dato entra ponetelo in un campo invertito poi alla fine visualizzatelo con caratteri normali. CURSOR LEFT è usato per ripristinare il programma, mentre RETURN indica la fine dell'ingresso dei dati. Ecco il programma completo:

```
80 REM CARICA IL NOME DI 20 CARATTERI
90 LN%=20
100 PRINT"[illegible]";TAB(10);
110 GOSUB 8000:NA$=CC$
120 REM CARICA LA VIA DI 20 CARATTERI
130 PRINT CHR$(13);TAB(10);
140 GOSUB 8000:SR$=CC$
150 REM CARICA LA CITTA' DI 20 CARATTERI
160 PRINT CHR$(13);TAB(10);
170 GOSUB 8000:CI$=CC$
180 REM CARICA LO STATO DI 20 CARATTERI
185 LN%=18
190 PRINT CHR$(13);TAB(10);
200 GOSUB 8000:ST$=CC$
210 REM CARICA IL CAP DI 5 CARATTERI
220 LN%=5
230 PRINT TAB(34);
240 GOSUB 8000:ZI$=CC$
250 STOP
8000 REM BATTERE UNA STRINGA LUNGA AL MASSIMO LN% CARATTERI
8010 REM IL CURSORE DEVE ESSERE ALL'INIZIO DEL CAMPO
8020 REM IL TASTO RETURN TERMINA L'INGRESSO NEL CAMPO
8030 REM IL TASTO ← FA RIPETERE L'INGRESSO NEL CAMPO
8040 REM NON VENGONO FATTI ALTRI CONTROLLI SU VALIDITA' DATI INGRESSO
8050 REM LA STRINGA IN INGRESSO VIENE RITORNATA COME CC$
8060 ST%=POS(X): REM PRENDE LA PRIMA POSIZIONE DEL CAMPO
8070 PRINT"[illegible]";:REM INVERSIONE CARATTERI
8080 FOR I=1 TO LN%: PRINT" ";:NEXT
8090 PRINT"[illegible]";CHR$(13);"[illegible]";TAB(ST%);
8100 REM INGRESSO E VISUALIZZAZIONE DEI DATI
8110 CC$="": J%=0
8120 FOR I=1 TO LN%
8125 J%=J%+1
8130 GET C$: IF C$="" THEN 8130
8140 IF C$="←" THEN PRINT CHR$(13);"[illegible]";TAB(ST%);:GOTO 8070
8150 IF C$=CHR$(13) THEN 8200
8160 PRINT C$;: CC$=CC$+C$
8170 NEXT
8190 REM COMPLETA CC$ CON SPAZI VUOTI
8200 IF J%=LN% THEN 8300
8210 FOR I=J% TO LN%
8220 CC$=CC$+" "
8230 NEXT
8300 PRINT CHR$(13);"[illegible]";TAB(ST%);CC$;
8310 RETURN
```

Caricate quindi questo programma.

Quando lo ponete in esecuzione ognuno dei cinque campi d'ingresso sarà reso più luminoso dall'inversione del campo. Alla fine, premendo RETURN, il campo dei caratteri diverrà normale.

Provate poi a premere CURSOR LEFT per far ripartire il programma.

Prima di procedere oltre esaminate attentamente la subroutine per l'ingresso dei dati che inizia alla linea 8060 e termina alla linea 8310 e cercate di capirne bene la struttura logica. Notate infine quanto sia facile per un operatore capire quali dati deve dare in ingresso ed eventualmente correggerli.

Vediamo ora un'altra funzione che deve essere svolta dal nostro programma di "data entry". Dopo che un intero nome ed indirizzo è stato dato il programma chiederà all'operatore se desidera fare qualche cambiamento e in tal caso quale campo deve essere modificato. Le subroutine per porre queste due domande le abbiamo già scritte all'inizio di questo capitolo. Qui di seguito riportiamo il programma completo con la nuova parte che inizia alla linea 250:

```
10 REM PROGRAMMA PER L'INGRESSO DI
11 REM NOMI E INDIRIZZI
20 REM VISUALIZZA IL MODULO D'INGRESSO
30 PRINT"IN BATTERE IL NOME E L'INDIRIZZO"
40 PRINT" 1■   NOME:"
50 PRINT" 2■    VIA:"
60 PRINT" 3■ CITTA':"
70 PRINT" 4■  STATO:";TAB(28);"5■ CAP:"
80 REM CARICA IL NOME DI 20 CARATTERI
90 LN%=20
100 PRINT"■■■";TAB(10);
110 GOSUB 8000:NA$=CC$
120 REM CARICA LA VIA DI 20 CARATTERI
130 PRINT CHR$(13);TAB(10);
140 GOSUB 8000:SR$=CC$
150 REM CARICA LA CITTA' DI 20 CARATTERI
160 PRINT CHR$(13);TAB(10);
170 GOSUB 8000:CI$=CC$
180 REM CARICA LO STATO DI 20 CARATTERI
185 LN%=18
190 PRINT CHR$(13);TAB(10);
200 GOSUB 8000:ST$=CC$
210 REM CARICA IL CAP DI 5 CARATTERI
220 LN%=5
230 PRINT TAB(34);
240 GOSUB 8000:ZI$=CC$
250 REM CHIEDE SE FARE DEI CAMBIAMENTI
260 QU$="VUOI FARE DEI CAMBIAMENTI? "
270 PRINT"■■■■■■■";
280 GOSUB 3000
290 IF YN$="N" THEN STOP
300 REM CHIEDE QUALE CAMPO CAMBIARE
310 QU$="BATTERE IL NUMERO (1-5) DI CAMPO DA CAMBIARE: "
320 LO%=1: HI%=5
330 GOSUB 3500
340 ON NM% GOTO 400,450,500,550,600,
400 REM CAMBIA IL NOME
410 PRINT"■■■";TAB(10);:LN%=20
420 GOSUB 8000:NA$=CC$
430 GOTO 260
450 REM CAMBIA LA VIA
460 PRINT"■■■■";TAB(10);:LN%=20
470 GOSUB 8000:SR$=CC$
480 GOTO 260
500 REM CAMBIA LA CITTA'
510 PRINT"■■■■■";TAB(10);:LN%=20
520 GOSUB 8000:CI$=CC$
530 GOTO 260
550 REM CAMBIA LO STATO
560 PRINT"■■■■■■";TAB(10);:LN%=18
570 GOSUB 8000:ST$=CC$
580 GOTO 260
600 REM CAMBIA IL CAP
610 PRINT"■■■■■■";TAB(34);:LN%=5
620 GOSUB 8000:ZI$=CC$
630 GOTO 260
2000 REM PULISCE LA RIGA SU CUI E' IL CURSORE
2010 PRINT CHR$(13);"■";: REM PORTA IL CURSORE SULLA COLONNA 0
2020 FOR I=1 TO 39:PRINT" ";:NEXT
```

```
2030 PRINT CHR$(13);"□";
2040 RETURN
3000 REM RISPONDERE ALLA DOMANDA CON Y O N PER YN$
3010 GOSUB 2000
3020 PRINT QU$;
3030 GET YN$: IF YN$<>"N" AND YN$<>"Y" THEN 3030
3040 PRINT YN$;
3050 RETURN
3500 REM CHIEDE UN VALORE NUMERICO CHE
3510 REM RITORNA CON NM%; NM% DEVE ESSERE
3520 REM COMPRESO TRA LO% E HI%
3530 REM IL PROGRAMMA CHIAMANTE DEVE DARE
3531 REM HI%,LO% E QU$
3540 GOSUB 2000
3550 PRINT QU$;
3560 GET NM$: IF NM$="" THEN 3560
3570 NM%=VAL(NM$)
3580 IF NM%<LO% OR NM%>HI% THEN 3560
3590 PRINT NM$;
3600 RETURN
8000 REM BATTERE UNA STRINGA LUNGA AL MASSIMO LN% CARATTERI
8010 REM IL CURSORE DEVE ESSERE ALL'INIZIO DEL CAMPO
8020 REM IL TASTO RETURN TERMINA L'INGRESSO NEL CAMPO
8030 REM IL TASTO ← FA RIPETERE L'INGRESSO NEL CAMPO
8040 REM NON VENGONO FATTI ALTRI CONTROLLI SU,VALIDITA' DATI INGRESSO
8050 REM LA STRINGA IN INGRESSO VIENE RITORNATA COME CC$
8060 ST%=POS(X): REM PRENDE LA PRIMA POSIZIONE DEL CAMPO
8070 PRINT"■";:REM INVERSIONE CARATTERI
8080 FOR I=1 TO LN%: PRINT" ";:NEXT
8090 PRINT"■";CHR$(13);"□";TAB(ST%);
8100 REM INGRESSO E VISUALIZZAZIONE DEI DATI
8110 CC$="": J%=0
8120 FOR I=1 TO LN%
8125 J%=J%+1
8130 GET C$: IF C$="" THEN 8130
8140 IF C$="←" THEN PRINT CHR$(13);"□";TAB(ST%);:GOTO 8070
8150 IF C$=CHR$(13) THEN 8200
8160 PRINT C$;: CC$=CC$+C$
8170 NEXT
8190 REM COMPLETA CC$ CON SPAZI VUOTI
8200 IF J%=LN% THEN 8300
8210 FOR I=J% TO LN%
8220 CC$=CC$+" "
8230 NEXT
8300 PRINT CHR$(13);"□";TAB(ST%);CC$;
8310 RETURN
```

Caricate ed eseguite questo programma; se qualche cosa non funziona vi diamo alcuni consigli per ricercare gli errori:

1. Se le visualizzazioni sullo schermo scorrono via in alto, allora vi sarete dimenticati di porre i punti e virgola in coda alle PRINT.
2. Se un campo inverso appare nel punto sbagliato, allora avete posto un numero errato di comandi CURSOR DOWN in una PRINT, oppure avete dimenticato di separare due variabili con il punto e virgola in una PRINT.
3. Se un messaggio non appare sul display allora controllate che le variabili che avete usato nel programma principale, per definire il testo, siano le stesse usate nelle subroutine.

Ripetiamo qui di seguito i punti fondamentali di questo programma:

1. Mediante il campo inverso d'ingresso indicate chiaramente all'operatore quali e quanti dati deve fornire e dove porli.

2. Quando l'operatore effettua una correzione il campo inverso lo avvisa dove essa è stata effettuata.
3. L'operatore non è obbligato a battere gli spazi bianchi perchè questi sono posti automaticamente.
4. In ogni momento l'operatore può ripetere l'ingresso in un campo premendo il tasto CURSOR LEFT.
5. Alle domande si deve rispondere con un carattere ben preciso come Y per SI' (YES) e N per NO. Rispondere ad una domanda con un carattere qualunque è molto pericoloso perchè rischiate di dare una risposta errata se accidentalmente tocate la tastiera.

Volendo avremmo potuto inserire nel nostro programma altre precauzioni come le seguenti:

1. Controllare che il codice CAP (ZIP negli USA) sia solo numerico.
2. Chiedere una conferma all'operatore quando le risposte sono negative. Questo per dare un'altra possibilità di correzione se per caso non si voleva dare la risposta NO.
3. Potremmo aggiungere un altro tasto speciale che rifiuti gli ultimi dati forniti e ripristini i precedenti. Infatti nel nostro programma, se per caso inseriamo una correzione su un campo invece esatto, siamo costretti a ricaricare il campo buono oltre che a ripetere la correzione che volevamo fare. Con questo tasto speciale possiamo subito rigettare l'ingresso errato e ripristinare il precedente valore.

Per esercizio provate ad aggiungere queste ulteriori opzioni. Se il vostro calcolatore è un CBM 8000 cercate di usare la sua TAB SET invece della funzione TAB.

PROGRAMMAZIONE DEL DISPLAY E DELLE STAMPANTI

Quando accendete il vostro calcolatore CBM l'uscita dei dati è mandata sul display. Se volete invece portare i dati su una stampante o su un'altra periferica dovete eseguire delle istruzioni particolari.

Anche se a prima vista può sembrare la stessa cosa, programmare un display o una stampante presenta delle notevoli differenze. Pensate per esempio che spesso una stampante ha un numero di colonne ben superiori a quello di un display; che sullo schermo del display potete muovere il cursore come volete, ma non potete di certo muovere liberamente la testina di una stampante.

Stante queste differenze la programmazione di un display e di una stampante sono molto simili.

Quanto diremo ora in questo paragrafo si riferisce solo ai display. Se volete scrivere programmi anche per stampanti dovete leggere, oltre a questo paragrafo, anche quanto vi diremo nel capitolo 6 per le stampanti.

Programmare l'uscita dei dati è molto più semplice di quanto si deve fare per il loro ingresso. E questo perchè non si deve tener conto dell'interazione con l'operatore.

Ecco alcune regole generali:

1. Evitare di ammassare troppa informazione in un piccolo spazio.
2. Se dei dati devono essere incolonnati, allora allineateli a destra o a sinistra così che siano di facile lettura.
3. Usate i caratteri in campo inverso sul display per porre in rilievo titoli o punti di riferimento del testo. Non usate il campo inverso per le stampanti perchè otterreste caratteri con pochissima risoluzione.

Vi riportiamo qui di seguito alcuni degli errori più comuni che si fanno quando si programmano dati in uscita:

1. Ricordatevi di porre tra le variabili dell'istruzione PRINT il punto e virgola (;) oppure la virgola (,).
2. Disegnate su un pezzo di carta la configurazione che deve assumere il testo di uscita sullo schermo prima di programmarlo. Risparmierete così molto tempo per calcolare la posizione delle colonne e delle righe e non dovrete fare inutili tentativi di esecuzione del vostro programma.
3. Se volete ripartire un vettore in un certo numero di colonne fate attenzione che la lunghezza del vettore sia divisibile per il numero di colonne. Per esempio supponete di avere un vettore N$(I) di lunghezza 25 che volete stampare su tre colonne. Potreste pensare di scrivere così:

```
100 FOR I=1 TO 25 STEP 3
200 REM PROCESS COLUMN 1
 :
 :
300 REM PROCESS COLUMN 2
 :
 :
400 REM PROCESS COLUMN 3
 :
 :
500 NEXT I
```

In tal caso calcolereste anche i valori 26 e 27 che non esistono. È sufficiente porre allora una istruzione di controllo della fine del vettore:

```
100 FOR I=LO TO HI STEP ST
 .
 .
 .
350 I=I+1
360 IF I>HI THEN 500
 .
 .
 .
500 NEXT
```

Una particolare attenzione la dovete prestare a quei dati che ottenete dalla lettura di un disco. (Per la gestione del disco vi rimandiamo al capitolo 6). I calcolatori CBM hanno l'antipatica abitudine di aggiungere spazi vuoti in coda alle stringhe che sono lette da disco. Per esempio se avete dei nomi con lunghezza non superiore a 20 caratteri e li avete scritti su un disco, potreste pensare che quando li leggete avrete ancora gli stessi nomi con la stessa lunghezza. Succede purtroppo che in coda ad alcuni saranno attaccati degli spazi vuoti. Questo vi può portare delle complicazioni per esempio facendovi superare un punto di tabulazione. Il problema può essere superato con la funzione LEFT$. Infatti se avete una PRINT di questo tipo:

```
100 PRINT TAB(5);N$(I);TAB(30);N$(I+1)
```

dovete riscriverla così:

```
100 PRINT TAB(5);LEFT$(N$(I),29);TAB(30);LEFT$(N$(I+1),20)
```

Se avete delle stringhe di lunghezza diversa vi può interessare di aggiungere spazi vuoti in coda alle più corte e troncare le più lunghe per averle tutte di eguale lunghezza. Ciò può essere fatto da questa subroutine:

```
10 REM LA VARIABILE DI STRINGA N$ DEVE ESSERE LUNGA 20 CARATTERI
20 REM SE E' PIU' CORTA VENGONO AGGIUNTI SPAZI VUOTI ANTERIORI
30 REM SE E' PIU' LUNGA VIENE TRONCATA
40 L%=LEN(N$): REM L%=NUMERO CARATTERI DI N$
50 B$="                    ": REM B$ CONTIENE 20 SPAZI VUOTI
60 IF L%>20 THEN N$=LEFT$(N$,20): REM N$ VIENE TRONCATA A 20 CARATT.
70 IF L%=20 THEN RETURN: REM N$ HA LA GIUSTA LUNGHEZZA
80 N$=N$+LEFT$(B$,20-L%): REM N$ CORTA: SI AGGIUNGONO SPAZI VUOTI
90 RETURN
```

Quando dovete trattare molti dati è consigliabile che definitiate una "finestra" in cui inserire i dati. Vi spieghiamo questo con un esempio. Supponiamo che X% sia una matrice intera di dimensione 14 per 50. Ogni suo elemento sia un numero intero di quattro cifre che rappresenta le sue coordinate nel modo seguente:

X%(I,J) = 0I0J

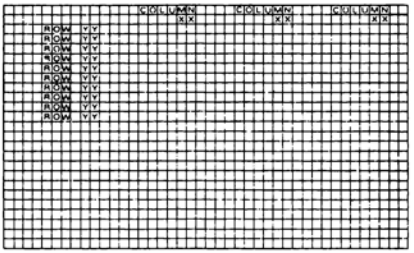
Per esempio:

X%(3,2) = 0302
X%(19,8) = 1908
X%(11,12) = 1112
etc.

Possiamo creare questa matrice con queste istruzioni:

```
10 DIM X%(14,50)
20 FOR I=1 TO 14
30 FOR J=1 TO 50
40 X%(I,J)=I*100+J
50 NEXT
60 NEXT
```

Ora desideriamo visualizzare una parte della matrice X%. Useremo le prime due righe e le prime 10 colonne per l'intestazione nel modo seguente:



XX rappresenta un numero tra 1 e 14
YY rappresenta un numero tra 1 e 50

L'intestazione è poi ottenuta in campo inverso con questo programma:

```
1000 REM CREA LE INTESTAZIONI DI RIGA  E DI COLONNA
1010 PRINT TAB(9);
1020 FOR I=1 TO 3
1030 PRINT"▒   COLONNA▒";
1040 NEXT
1050 PRINT CHR$(13);TAB(9);
1060 FOR I=C% TO C%+2
1070 S%=7: IF I<10 THEN S%=8
1080 PRINT SPC(S%);"▒";STR$(I);"▒";
1090 NEXT
1095 PRINT CHR$(13);
1110 FOR I=R% TO R%+9
1120 S%=1: IF I<10 THEN S%=2
1130 PRINT TAB(1);"▒RIGA";SPC(S%);STR$(I);"▒"
1140 NEXT
1150 RETURN
```

Abbiamo deliberatamente creato una finestrella più piccola di tutto lo schermo per meglio descrivere il concetto di finestra di dati. Se lo ritenete opportuno potete anche estenderla a tutto lo schermo.

La stringa STR$ è più lunga di un carattere del numero intero per poterne rappresentare il segno. Esso sarà visualizzato in campo inverso e bisogna tenere conto della sua presenza alla linea 1130 dove si definiscono le posizioni di tabulazione.

Aggiungiamo ora le istruzioni per chiedere all'operatore quali sono i valori minimi di riga e colonna che vuole visualizzare. Questo elemento della matrice apparirà in alto a sinistra e la finestra sarà completata con gli elementi successivi. Ecco le istruzioni da aggiungere:

```
5 REM CREAZIONE DI UNA FINESTRA
10 DIM X%(14,50)
20 FOR I=1 TO 14
30 FOR J=1 TO 50
40 X%(I,J)=I*100+J
50 NEXT
60 NEXT
```

```
64 PRINT"▒";
65 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒";
70 INPUT"DARE LA COL. (1-12)";CN
80 IF CN<1 OR CN>12 THEN PRINT"▒";: GOTO 70
90 INPUT"DARE LA RIGA (1-41)";RN
100 IF RN<1 OR RN>41 THEN PRINT"▒";: GOTO 90
105 PRINT"▒";: GOSUB 1000
110 PRINT"▒▒▒";
120 FOR I=RN TO RN+9
130 PRINT TAB(9);
140 FOR J=CN TO CN+2
150 N$=STR$(N%(J,I))
155 PRINT SPC(10-LEN(N$));N$;
160 NEXT
165 PRINT CHR$(13);
170 NEXT
180 PRINT"▒▒▒CONTINUI? BATTERE Y O N "
190 GET C$: IF C$<>"Y" AND C$<>"N" THEN 190
200 IF C$="Y" THEN 65
210 STOP
```

Eseguite il programma. Per 10 o 15 secondi vi sembrerà che il calcolatore non faccia niente, ma in realtà sta riempiendo la matrice X$ mediante il ciclo FOR NEXT dalla linea 20 alla 60.

La PRINT della linea 64 pulisce lo schermo. Le INPUT delle linee 70 e 90 chiedono quali sono i valori minimi della riga e della colonna. Il comando per pulire lo schermo non è stato posto alla linea 65 perchè il programma ritorna appunto a questa linea e non vogliamo che la visualizzazione precedente venga cancellata.

Notate che in ingresso viene chiesto il numero di colonna al massimo fino a 12 perchè le altre due sarebbero in ogni caso visualizzate. Analogamente il massimo numero per le righe è 41 perchè in tal caso vengono visualizzate le righe da 41 a 50.

I valori interi della matrice X% sono convertiti in stringhe alla linea 150 prima di essere stampati alla linea 155. Questa conversione in stringhe ci permette di rendere più semplice la stampa e l'allineamento dei valori. Per provare questo rimuovete la linea 150 e cambiate come segue la 155:

```
155 PRINT SPC(5);N%(J,I);
```

I numeri saranno allineati purchè non visualizzate alcun numero di quattro cifre; in tal caso le 40 colonne dello schermo saranno superate. Se visualizzate numeri con tre cifre le righe saranno spostate a destra di una colonna. Potete correggere questa anomalia aumentando la tabulazione della linea 130 da 9 a 10. Quando porrete in esecuzione il programma le righe a 40 colonne saranno superate (overflow) e avrete molti ritorni a capo.

Notate che dopo le istruzioni d'ingresso alle linee 70, 90 e 180 sono stati posti dei controlli sulla validità dei dati forniti.

Quando si usa una finestra è molto comodo permettere all'operatore di spostare a piacimento la finestra stessa. Con il simbolo di picche (♠) stabiliamo di muoverla

di una riga in alto, cioè che il numero iniziale delle righe sia diminuito di uno. Con il simbolo di cuori(♥)di muoverla invece di una riga in basso, cioè che il valore iniziale sia incrementato di uno. Con il simbolo di minore (<) di spostarla di una colonna a sinistra, cioè che il valore iniziale delle colonne sia diminuito di uno. E infine stabiliamo che con il segno di maggiore (>) la finestra sia spostata a destra di una colonna e quindi il valore d'ingresso delle colonne sia incrementato di uno.

Il programma per ottenere questo è il seguente e richiede di sostituire le linee da 180 a 210:

```
180 PRINT"■■■CONTINUI? BATTERE ↑,♥,<,>,Y O N ";
190 GET C$: IF C$="" THEN 190
200 REM SE C$=PICCHE ALLORA MUOVE DI UNA RIGA IN ALTO
210 IF C$="↑" THEN R%=R%-1: PRINT CHR$(13);"■";: GOTO 100
220 REM SE C$=CUORI ALLORA MUOVE DI UNA RIGA IN BASSO
230 IF C$="♥" THEN R%=R%+1: PRINT CHR$(13);"■";: GOTO 100
240 REM SE C$=< ALLORA MUOVE DI UNA COLONNA A SINISTRA
250 IF C$="<" THEN C%=C%-1: GOTO 300
260 REM SE C$=> ALLORA MUOVE DI UNA COLONNA A DESTRA
270 IF C$=">" THEN C%=C%+1: GOTO 300
280 REM SE C$=Y, BATTERE NUOVA RIGA E COLONNA; SE C$=N ALLORA STOP
290 IF C$="Y" THEN 65
295 IF C$="N" THEN STOP
296 GOTO 190: REM RIGETTA OGNI ALTRO VALORE C$
300 IF C%<1 OR C%>12 THEN PRINT CHR$(13);: GOTO 70
310 GOTO 105
```

Notate come è affidabile la logica di questo programma. Ogni ingresso diverso dai sei consentiti viene subito rigettato. Se cambiando i numeri delle righe o delle colonne essi escono dai limiti consentiti, il programma chiede semplicemente di dare nuovi valori. (Le funzioni finestra e "scrolling" del calcolatore CBM non sono molto utili in questo esempio poichè noi vogliamo muovere la finestra a destra e a sinistra così come la muoviamo in alto e in basso).

Un aspetto poco elegante di questo programma riguarda il fatto che superando il valore consentito per le righe, è possibile ridare solo il nuovo valore per le righe stesse. Questo dipende dall'istruzione GOTO 100 delle linee 210 e 230. Se invece viene superato il valore delle colonne è possibile ridare sia il valore delle righe che delle colonne, come si può vedere con la GOTO 70 della linea 300. Provate voi a correggere questa anomalia facendo in modo che si possano dare sempre ambedue i valori di riga e colonna oppure uno solo.

Un altro aspetto che può essere migliorato riguarda il tempo che il calcolatore impiega a riempire il vettore X%. Un operatore potrebbe erroneamente pensare che qualcosa non funzioni mentre invece il calcolatore sta compiendo il lungo ciclo di caricamento di X%. È consigliabile quindi far apparire un messaggio di avviso sul display, prima che inizi tale ciclo, il cui contenuto potrebbe essere il seguente:

```
15 PRINT "■ ATTENDERE!  CARICAMENTO DELLA MATRICE X%"
```

Notate che il nostro programma pone molta attenzione a far terminare le visualizzazioni alla 39-esima colonna anzichè alla 40-esima. **Quando usate un calcolatore CBM a 40 colonne è preferibile che non superiate la colonna 40 perchè**

correte il rischio di essere coinvolti nella tecnica di continuazione delle righe detta ad "avvolgimento" (wrap around). In tal caso una riga successiva è vista come la continuazione di una riga precedente e vi sarebbe una gran confusione tra i ritorni a capo da voi stabiliti e quelli forzati dalla tecnica di "wrap around".

Tecnica di "wrap around" per schermo a 40 colonne

Quando il cursore si trova sulla 40-esima colonna di una qualunque riga e voi battete appunto il 40-esimo carattere di quella riga, il cursore va a capo assumendo però quella posizione come 41-esima di una riga a 80 colonne. Cioè la seconda riga dello schermo è vista come seconda metà di una riga più lunga.

Con la tecnica di "wrap around", quando date il comando RETURN, il calcolatore esegue un ritorno a capo alla successiva riga logica. In altre parole è il calcolatore che riconosce se una riga logica può stare su una sola riga fisica dello schermo e lui stesso determina quindi il giusto salto di riga al momento del ritorno a capo.

Se usate il comando POKE per scrivere sulla 40-esima colonna di uno schermo a 40 colonne, allora il calcolatore non assume di potere avere righe di 80 caratteri. Questo può essere fatto con la seguente istruzione:

```
POKE 32767 + (L-1)* 40,ASC (CH$)
```

dove:

L è il numero della linea
CH$ è il carattere da forzare con POKE

PROGRAMMAZIONE MATEMATICA

I calcolatori CBM possono effettuare le operazioni di addizione, sottrazione, moltiplicazione e divisione su numeri rappresentati con nove cifre. Un numero di nove cifre può però non essere sufficiente e causare quindi problemi di calcolo. Per esempio il campo consentito per i numeri interi va da 0 a 999999999. Mentre quello consentito per i numeri frazionari, o con virgola, va da 0.000000001 a 99999999.9. Per molti calcoli questi numeri sono sufficienti, ma vi sono molti problemi di ingegneria o commerciali che richiedono numeri con un maggior numero di cifre.

Per superare questa limitazione esistono due metodi di programmazione: il primo usa le stringhe numeriche e il secondo la matematica dei multipli interi in cui un numero grande viene ripartito in segmenti più piccoli e ogni segmento è trattato separatamente.

ADDIZIONE

Sia le stringhe intere che la tecnica dei multipli interi possono essere usate per addizionare numeri con più di 9 cifre. "Augendo" è il primo numero dell'equazione e "addendo" il secondo. L'addendo è sommato all'augendo.

Addizione mediante stringhe numeriche

I passi da seguire sono:

1. Fate entrare l'augendo e l'addendo come due stringhe numeriche positive.
2. Allineate a destra le stringhe.
3. Addizionate le cifre corrispondenti delle due stringhe compreso il riporto.
4. Concatenate il risultato in una unica stringa.
5. Stampate la risposta.

Passo 1: Fate entrare l'augendo e l'addendo come due stringhe numeriche positive mediante una istruzione INPUT.

Sullo schermo

```
10 PRINT"❑***ADDITION***":PRINT
20 INPUT A$,B$

RUN

***ADDITION***

?1234567890123456
??57943572
```

In memoria

```
A$ 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
B$ 5 7 9 4 3 5 7 2
```

A$ è l'augento e B$ l'addendo. Con l'istruzione INPUT potete superare il limite delle nove cifre per numero. In questo esempio consideriamo solo numeri positivi, ma potete benissimo estendere il programma anche a quelli negativi e ai frazionari.

Passo 2: Allineate a destra le stringhe. Prima di effettuare qualunque calcolo aritmetico dovete allineare le stringhe a destra in quanto il BASIC CBM le allinea invece a sinistra. Se non lo fate otterrete risultati completamente errati; infatti guardate questo esempio:

Allineamento a sinistra - errato

```
  1234567890123456
 +57943572
  7028925090123456
```

Allineamento a destra - corretto

```
  1234567890123456
 +        57943572
  1234565948067028
```

Le istruzioni seguenti allineano a destra le più corta delle due stringhe riempiendola di zeri a sinistra. Si ottengono così due stringhe di lunghezza uguale. X è la lunghezza di A$ e Y quella di B$:

```
30 BLANK$="                    "
40 X=LEN(A$):Y=LEN(B$)
50 IF X<Y THEN A$=LEFT$(BLANK$,Y-X)+A$
60 IF X>Y THEN B$=LEFT$(BLANK$,X-Y)+B$
```

Abbiamo assunto di trattare in questo esempio numeri con 16 cifre al massimo. Per questo motivo alla linea 30 abbiamo preparato una stringa buffer BLANK$ contenente 16 spazi vuoti.

Alle linee 50 e 60 viene ricercata la stringa più corta (nel nostro caso B$) e la sua lunghezza sottratta a quella dell'altra stringa:

```
60 IF Y<X THEN B$=LEFT$(BLANK$,X-Y)+B$
```

Lunghezza della stringa più corta sottratta a quella della stringa più lunga

Se per esempio la lunghezza di A$ fosse 16 e quella di B$ 8, la differenza sarebbe ancora 8:

A$	1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6	X = 16	X - Y = 8
B$	5 7 9 4 3 5 7 2	Y = 8	16 - 8 = 8

A questo punto inseriamo 8 spazi vuoti davanti a B$ per ottenere anche B$ lungo 16 cifre; l'istruzione è la seguente:

```
LEFT$(BLANK$,X-Y)+B$
```

e al procedura è la seguente:

```
B$=LEFT$(BLANK$,X-Y)                    +B$
B$=LEFT$(BLANK$,16-8)                   +B$
B$=LEFT$(BLANK$,8)                      +B$
B$=LEFT$(ØØØØØØØØØØØØØØØØ,8)            +B$
B$=ØØØØØØØØ                             +5 7 9 4 3 5 7 2
B$=ØØØØØØØØ5 7 9 4 3 5 7 2
```

A$= 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 — **16 cifre**

B$= ØØØØØØØØ5 7 9 4 3 5 7 2 — **16 cifre**

Passo 3: Addizionate le cifre corrispondenti delle due stringhe. Potreste pensare che la somma delle due stringhe si possa fare così:

```
C$=A$+B$
```

Questo è un notevole errore perchè così si concatenano le stringhe e non si sommano:

```
C$=A$+B$
C$= 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6  +  ØØØØØØØØ5 7 9 4 3 5 7 2
C$= 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6ØØØØØØØØØØØØØØØØ5 7 9 4 3 5 7 2
```

Per sommare il contenuto numerico delle due stringhe bisogna estrarre cifra per cifra trasformarle in numero e quindi sommarle. Per fare questo useremo le funzioni VAL e MID$:

```
1020 FOR I=LEN(A$) TO 1 STEP-1
1030 A=VAL(MID$(A$,I,1))
1050 B=VAL(MID$(B$,I,1))
1100 NEXT I
```

Indichiamo con A la cifra estratta da A$ e con B quella estratta da B$. I è invece un contatore inizializzato con il valore della lunghezza massima delle due stringhe. Nel ciclo FOR NEXT I viene decrementato e una cifra alla volta viene estratta da destra verso sinistra con la funzione MID$:

```
1        MID$(B$,I,1)

16  ØØØØØØØØ5794357 2
15  ØØØØØØØØ5794357 2
14  ØØØØØØØØ5794357 2
13  ØØØØØØØØ5794357 2
12  ØØØØØØØØ5794357 2
11  ØØØØØØØØ5794357 2
10  ØØØØØØØØ5794357 2
 9  ØØØØØØØØ5794357 2
 8  ØØØØØØØØ5794357 2
 7  ØØØØØØØØ5794357 2
 6  ØØØØØØØØ5794357 2
 5  ØØØØØØØØ5794357 2
 4  ØØØØØØØØ5794357 2
 3  ØØØØØØØØ5794357 2
 2  ØØØØØØØØ5794357 2
 1  ØØØØØØØØ5794357 2
```

La funzione VAL converte poi ogni cifra nel suo valore numerico:

Quando I = 16,

```
B=VAL(MID$(B$,16,1))
B=VAL($ØØØØØØØØ5794357 2)
B= Ø2
```

Quando I = 15,

```
B=VAL(MID$(B$,15,1))
```

Man mano che le cifre sono convertite in numero vengono sommate e il risultato riportato nella stringa C$:

```
1000 N=1                                  Inizializza il puntatore di stringa N
1010 D=0                                  Inizializza il riporto
1020 FOR I=LEN(A$) TO 1 STEP -1           Inizializza il contatore di decremento I
1030 A=VAL(MID$(A$,I,1))                  Estrae le cifre separatamente e le converte in numerico
1040 A=A+D:D=0                            Addiziona il riporto D; azzera il riporto D
1050 B=VAL(MID$(B$,I,1))                  Estrae le cifre separatamente e le converte in numerico
1060 C=A+B                                Addiziona i valori estratti di A$ e B$
1070 IF C>=10 THEN D=1                    Tiene conto del riporto
1080 IF D=1 AND I=1 THEN N=2
1090 C$=RIGHT$(STR$(C),N)+C$              Forma la stringa risultato
1100 NEXT I
```

Ovviamente bisogna tenere conto del riporto, che eventualmente otteniamo sommando ogni coppia di cifre, per cui introduciamo la variabile D che conterrà appunto tale riporto. Inizialmente D è posto eguale a zero alla linea 1010. Durante le singole somme se C diviene maggiore o eguale a 10 allora D è posto eguale a 1 per essere poi sommato con le successive due cifre:

```
    1 2 3 +1 4 +1 5 +1 6  1 7 +1 8 9 0 1 2
+             5  7  9  4 3 5 7 2
  ---------------------------------------
    1 2 3  5  1  4  7  3 2 5 8 4
```

Alla linea 1070 il riporto viene posto eguale a 1 se C è maggiore o eguale a 10; altrimenti esso rimane nullo:

```
1070 IF C>=10 THEN D=1
 A   [06]
+B   [09]
 ---------
 C   [015] ---> 15>=10 -> D[01]

or
 A   [03]
+B   [00]
 ---------
 C   [03] ---> 3<10 -> D[00] (nessun cambiamento)
```

Notate che D al massimo può essere unitario perchè stiamo sommando solo due cifre la cui somma massima è 18 con riporto di 1.

Alla linea 1040 viene fatta la somma del riporto e in tal caso D è azzerato.

Passo 4: Concatenate il risultato in una unica stringa. Anche il risultato dovrà essere espresso sotto forma di stringa per ovviare alla limitazione delle nove cifre massime.

Alla linea 1090 viene costruita la stringa risultato C$. La funzione STR$(C) converte C in stringa. La funzione RIGHT$ estrae gli N caratteri più a destra di STR$(C). Normalmente N è posto eguale a 1 (linea 1000) perchè vogliamo estrarre solo la cifra significativa e non il segno di ogni singola somma. Il carattere più a sinistra di C rappresenta invece il segno e non vogliamo che sia ogni volta concatenato in C$.

```
1000 N=1
     N[01]
1060 C=A+B
     C[08] = A[06] + B[02]
1090 C$=RIGHT$(STR$(C),N)+C$
     C$=RIGHT$(STR$(C),1)+C$
     C$=RIGHT$([08],1)+C$
     C$=[8] + C$
```

Anche quando C supera 10 noi dobbiamo mantenere solo la cifra più a destra perchè l'altra è già compresa nel riporto.

N verrà posto eguale a 2 solo nel caso che l'ultimo riporto sia unitario D=1 e che il contatore I sia I=1. Questo è importante perchè, se ambedue queste condizioni sono vere, il ciclo non dovrà iterare ancora ed eseguire la linea 1040 dove D perde il suo valore. Ponendo N=2 nell'ultimo ciclo ambedue le cifre di C sono poste in C$ e l'ultimo riporto non viene quindi perso.

```
1070 IF C>=10 THEN D=1
          C[012] >=10     D[01]
1080 IF D=1 AND I=1 THEN N=2
          D[01]        I[01]        N[02]
1090 C$=RIGHT$(STR$(C),N)+C$
     C$=RIGHT$([012],2)+C$
     C$=[12]+C$
     C$=[12XXXXXXXX]
```

Ricapitoliamo le funzioni svolte dal ciclo FOR NEXT dalla linea 1020 alla linea 1100:

1. Estrazione delle singole cifre dalle due stringhe numeriche e loro conversione in numero (istituzioni 1030 e 1050).

2. Le cifre corrispondenti sono sommate due alla volta (istruzione 1060) e viene posto l'eventuale riporto (istruzione 1070). Il riporto è sommato ad A nella colonna successiva (1040).
3. Le singole somme sono quindi unite e convertite in unica stringa (1090).

Passo 5: Stampate la risposta. Per completare questa routine di addizione inseriamo le istruzioni di ingresso e di misura delle lunghezze delle stringhe (istruzioni da 10 a 1010). Alla fine poniamo le istruzioni di uscita e di azzeramento (1110 e 1120). Il programma completo si presenta così:

```
10 PRINT"⊃***ADDITITON***" PRINT        Pulisce lo schermo
20 INPUT A$,B$                          Ingresso stringhe numeriche
30 BLANK$="
40 X=LEN(A$):Y=LEN(B$)
50 IF X<Y THEN A$=LEFT$(BLANK$,Y-X)+A$  } Allineamento a destra
60 IF Y<X THEN B$=LEFT$(BLANK$,X-Y)+B$  }
1000 N=1
1010 D=0
1020 FOR I=LEN(A$) TO 1 STEP-1          }
1030 A=VAL(MID$(A$,I,1))                }
1040 A=A+D:D=0                          }
1050 B=VAL(MID$(B$,I,1))                } Ciclo di addizione
1060 C=A+B                              }
1070 IF C>=10 THEN D=1                  }
1080 IF D=1 AND I=1 THEN N=2            }
1090 C$=RIGHT$(STR$(C),N)+C$            }
1100 NEXT I                             }
1110 PRINT:PRINT"ANSWER= ";C$           Stampa C$
1120 C$="":PRINT:GOTO 20                Azzera C$
1130 END
```

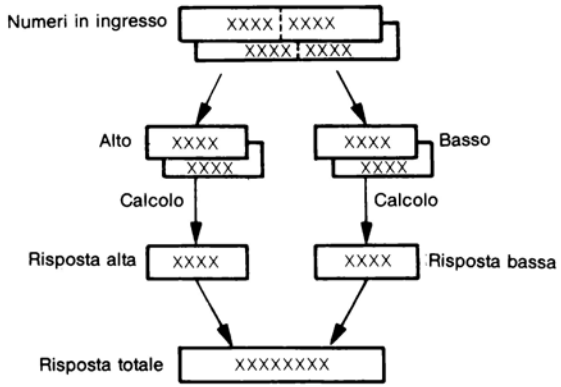
Ecco un esempio di esecuzione:

```
***ADDITION***

?12345
??579

ANSWER=  12924

?1234567890123456
??57943572

ANSWER=  1234567948067028
```

Addizione di interi multipli

Un altro metodo per superare l'ostacolo delle nove cifre massime è quello che fa uso degli interi multipli.

La tecnica degli interi multipli ripartisce un numero grande in segmenti più piccoli che vengono poi trattati separatamente. Le singole risposte sono poi unite in un unico risultato:



I passi di questa procedura sono i seguenti:

1. Caricate l'augendo e l'addendo come due stringhe numeriche positive.
2. Dividete i numeri in due parti eguali che verranno chiamate alta e bassa.
3. Calcolate separatamente la somma delle parti alte e delle parti basse.
4. Concatenate i due risultati in una unica stringa.
5. Visualizzate il risultato.

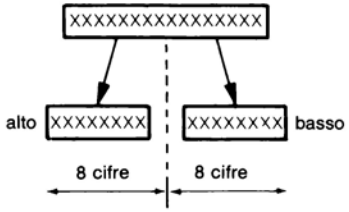
Passo 1: Caricate l'augendo e l'addendo come due stringhe numeriche positive:

```
10 PRINT"❑***MULTIPLE INTEGER ADDITION***":PRINT
20 INPUT A$,B$

RUN

***MULTIPLE INTEGER ADDITION***

?1234567890123456
??757943572
```

A$ è l'augendo e B$ è l'addendo. I due numeri sono caricati come stringhe per ovviare alla limitazione delle nove cifre massime e per poter usare le funzioni di stringa per dividere i due numeri in parti più piccole.

Passo 2: Determinare la massima lunghezza dei numeri in ingresso e calcolare il numero di segmenti in cui ripartirli. Per esempio se la massima lunghezza è 16 i due numeri possono essere ripartiti in due parti di 8 cifre.

Per rendere più semplice il nostro esempio supponiamo che la lunghezza massima sia appunto 16 e divideremo i nostri numeri nella parte alta e parte bassa di 8 cifre ciascuna.



Dapprima dobbiamo determinare quale numero è il più lungo. X e Y sono le lunghezze di A$ e B$ rispettivamente:

```
1000 X=LEN(A$) Y=LEN(B$)
```

Successivamente si confrontano le due lunghezze. Se X > Y allora la variabile F (cioè il fattore di divisione) è posta eguale a un mezzo di X. Se invece Y > X allora F è posta eguale a un mezzo di Y:

```
1002 IF X>Y THEN F=X/2 GOTO 1006
1004 F=Y/2
```

oppure in quest'altro modo:

```
1002 F=Y/2:IF X>Y THEN F=X/2
```

Nel nostro esempio A$ = "1234567890123456" e B$ = "57943572" per cui otteniamo:

```
1000 X=LEN(A$) Y=LEN(B$)
     X = 16         Y = 8
1002 IF X>Y THEN F=X/2:GOTO 1006
     16>8 Istruzione vera, quindi
          F = 16/2
          F = 8
          Il programma continua alla linea 1006
```

Una volta calcolato il valore di F il programma continua alla linea 1006. Nel caso che F sia un numero frazionario allora esso viene posto eguale alla sua parte intera più uno. Per esempio se avessimo trovato per F 7.5 allora si pone F = 8:

```
1006 IF F>INT(F) THEN F=INT(F)+1
     Se 7.5 > 7 allora F = 7 + 1
                       F = 8
```

Per ottenere poi la parte alta (High) e la parte bassa (Low) della somma di A$ e B$ ecco qui di seguito le istruzioni:

```
1000 X=LEN(A$):Y=LEN(B$)
1002 IF X>Y THEN F=X/2:GOTO 1006
1004 F=Y/2
1006 IF F>INT(F) THEN F=INT(F)+1
1010 IF X<=F THEN AH=0:AL=VAL(A$):GOTO 1040
1020 AH=VAL(LEFT$(A$,X-F))
1030 AL=VAL(RIGHT$(A$,F))
1040 IF Y<=F THEN BH=0:BL=VAL(B$):GOTO 1070
1050 BH=VAL(LEFT$(B$,Y-F))
1060 BL=VAL(RIGHT$(B$,F))
```

Le istruzioni dalla linea 1010 alla 1040 confrontano le lunghezze delle stringhe con il divisore F che in questo caso è 8. Se la stringa è minore di 8, allora a AH (o BH) viene assegnato il valore zero lasciando AL eguale a A$ (o BL eguale a B$). Se la stringa è più lunga di 8 deve essere divisa nella parte alta e in quella bassa. Le parti basse AL o BL ricevono i valori corrispondenti alle 8 cifre poste a destra, mentre le parti alte AH e BH ricevono quelli posti a sinistra delle parti basse:

```
1020 AH=VAL(LEFT$(A$,X-F))
     AH=VAL(LEFT$(A$,16-8))
     AH=VAL(LEFT$(1234567890123456,8))
     AH=VAL(12345678)
     AH=12345678
1030 AL=VAL(RIGHT$(A$,F))
     AL=VAL(RIGHT$(1234567890123456,8))
     AL=VAL(90123456)
     AL=90123456
```

E analogamente per BH e BL. **Ricordiamo che la funzione VAL converte una stringa numerica in un numero.**

Passo 3: Una volta che le stringhe sono state divise in segmenti sufficientemente piccoli, così che il calcolatore CBM li possa trattare direttamente, allora inizia la procedura di addizione. Con questa tecnica addizionate gruppi di numeri corrispondenti. AH sarà addizionato a BH e AL a BL. A questo punto possiamo sfruttare le capacità aritmetiche del nostro calcolatore perchè si tratta di sommare numeri contenuti nei limiti delle nove cifre. Nel caso precedente dovevamo invece fare la somma cifra per cifra in quanto non potevamo addizionare numeri rappresentati da stringhe.

```
  AH  12345678        AL  90123456
 +BH  00000000       +BL  57943572
 ------------        -------------
  CH  12345678        CL  148067028
```

Le parti basse possono essere così sommate:

```
1070 CL$=STR$(AL+BL)
```

e poste subito come parte bassa del risultato CL$. Non è necesario che il risultato di questa somma parziale sia posto sotto forma di stringa, ma questo permette di usare la funzione LEN per tener conto del riporto.

Lo spazio riservato al segno di CL$ viene troncato mediante la funzione MID$ in quanto non dobbiamo conservarlo per ottenere la stringa finale C$; questo è fatto alla linea 1075.

Alla linea 1080 viene fatto il controllo se esiste il riporto. Ciò si può fare vedendo se la lunghezza di CL$ è maggiore di F; in tal caso la cifra più a sinistra sarà sommata alla seconda somma parziale CH$. (Il valore del riporto D può essere 0 o 1).

CH$ è ottenuta sommando AH, BH e D.

```
1070 CL$=STR$(AL+BL)
     CL$=STR$(ø90123456 + ø57943572)
     CL$=STR$(ø148067028)
     CL$=ø148067028
1075 CL$=MID$(CL$,2,LEN(CL$)-1)
     CL$=MID$(ø148067028,2,10-1)
     CL$=MID$(ø148067028,2,9)
     CL$=148067028
1080 IF LEN(CL$)>F THEN D=1
     LEN(CL$)=9  F=8
     9>8→D=1
1090 CH$=STR$(AH+BH+D)
     CH$=STR$(ø12345678 + ø00000000 + ø1)
     CH$=STR$(ø12345679)
     CH$=ø12345679
1095 CH$=MID$(CH$,2,LEN(CH$)-1)
     CH$=MID$(ø12345679,2,10-1)
     CH$=MID$(ø12345679,2,9)
     CH$=12345679
```

Passo 4: Concateniamo ora le due somme parziali CH$ e CL$. Il segno e il riporto sono troncati da CL$ prendendo le sole 8 cifre più a destra della stringa:

```
1100 C$=CH$+RIGHT$(CL$,F)
     C$=CH$(Ø123456789 + RIGHT$(CL$(Ø148067028,8)
     C$=Ø123456789 + 48067028
     C$=Ø123456794806702 8
```

Passo 5: Visualizzazione del risultato.

```
1110 PRINT:PRINT"ANSWER=";C$:PRINT
```

Il programma è ora completo. Questa addizione di interi multipli accetta due numeri interi positivi che possono avere fino a 16 cifre. Ogni numero è diviso in due segmenti, uno alto e uno basso, di 8 cifre ciascuno. I segmenti alti e quelli bassi sono sommati separatamente e i due risultati sono concatenati in una unica stringa che può essere lunga sino a 17 cifre. In altre parole questa routine vi permette di avere 8 cifre di più del massimo consentito per i calcolatori CBM.

Ecco il programma completo e un suo esempio di esecuzione:

```
10 PRINT"❑***MULTIPLE INTEGER ADDITITON***":PRINT
20 INPUT A$,B$
1000 X=LEN(A$):Y=LEN(B$)
1002 IF X>Y THEN F=X/2:GOTO 1006
1004 F=Y/2
1006 IF F>INT(F) THEN F=INT(F)+1
1010 IF X<=F THEN AH=0:AL=VAL(A$):GOTO 1040
1020 AH=VAL(LEFT$(A$,X-F))
1030 AL=VAL(RIGHT$(A$,F))
1040 IF Y<=F THEN BH=0:BL=VAL(B$):GOTO 1070
1050 BH=VAL(LEFT$(B$,Y-F))
1060 BL=VAL(RIGHT$(B$,F))
1070 CL$=STR$(AL+BL)
1075 CL$=MID$(CL$,2,LEN(CL$)-1)
1080 IF LEN(CL$)>F THEN D=1
1090 CH$=STR$(AH+BH+D)
1095 CH$=MID$(CH$,2,LEN(CH$)-1)
1100 C$=CH$+CL$
1110 PRINT:PRINT"ANSWER=";C$:PRINT
1120 AH=0:AL=0:BH=0:BL=0:D=0:CH$="":CL$="":C$="":GOTO 20
1130 END

***MULTIPLE INTEGER ADDITION***

?1234567890123456
??57943572

ANSWER= 1234567948067028
```

SOTTRAZIONE

In analogia a quanto visto per l'addizione, potete sottrarre numeri con più di nove cifre usando uno dei due metodi: mediante stringhe numeriche o con l'uso degli interi multipli.

Sottrazione mediante stringhe numeriche

Molte delle cose dette per l'addizione valgono anche per la sottrazione. I passi della procedura per la sottrazione sono i seguenti:

1. Caricate il minuendo ed il sottraendo come due stringhe numeriche positive.
2. Allineate a destra le due stringhe.
3. Determinate quale è la stringa più lunga.
4. Sottraete le cifre corrispondenti delle due stringhe tenendo conto dei riporti a prestito.
5. Concatenate i risultati parziali in una unica stringa.
6. Eliminate gli zeri posti avanti al risultato.
7. Visualizzate il risultato.

Passo 1: Mediante una istruzione INPUT **caricate i valori del minuendo e del sottraendo** come due stringhe numeriche positive:

```
10 PRINT"***SUBTRACTION***":PRINT
20 INPUT A$,B$

RUN

***SUBTRACTION***

?123456789012                    A$ 1 2 3 4 5 6 7 8 9 0 1 2
??57943572                       B$ 5 7 9 4 3 5 7 2
```

A$ è il minuendo (il numero a cui sarà tolto un altro numero).
B$ è il sottraendo (il numero tolto al minuendo).

Passo 2: Allineate a destra le due stringhe numeriche come avete già visto fare per l'addizione:

```
30 BLANK$="                                  "
40 X=LEN(A$):Y=LEN(B$)
50 IF X<Y THEN A$=LEFT$(BLANK$,Y-X)+A$
60 IF Y<X THEN B$=LEFT$(BLANK$,X-Y)+B$
```

Passo 3: Nel caso della sottrazione dobbiamo prima **determinare quale delle due stringhe numeriche ha il valore maggiore.** Sebbene due stringhe possano avere eguale lunghezza, i loro valori possono essere molto diversi.

I valori di A$ e di B$ sono confrontati mediante la funzione VAL alle linee 65 e 70:

```
65 IF VAL(A$)=VAL(B$) THEN C$="0":GOTO 1150
70 IF VAL(A$)>VAL(B$) GOTO 1000
```

Noi supponiamo di sottrarre B$ da A$.

Se A$ è maggiore di B$ abbiamo un semplice problema di sottrazione e il

programma prosegue alla linea 1000. Se invece B$ è maggiore di A$ avremo un risultato negativo.

In questo secondo caso cambiamo il contenuto di A$ con B$ e viceversa, così da sottrarre sempre il numero minore dal maggiore, e poi poniamo il segno meno al risultato.

Per esempio sottraiamo 5 da 3. È ovvio che in tal caso VAL (B$) > VAL (A$) cioè il sottraendo è maggiore del minuendo.

A$ [3]
B$ [5]
Cambiare A$ con B$:
A$ [5] B$ [3] → A$ [5] B$ [3]
Sottrarre: VAL(A$) − VAL(B$) = C$
A$ [5] − B$ [3] → C$ [2]
Convertire in negativo:
C$ = "−" + C$
"−" + C$ [2] → C$ [−2]
Risposta:
C$ [−2]

Le due variabili sono commutate alla linea 80:

```
80 X$=A$:A$=B$:B$=X$
```

Istruzione	Memoria		
	X$	A$	B$
⋮	0	3	5
X$=A$	3	3	5
A$=B$	3	5	5
B$=X$	3	5	3

X$ ha la funzione di memoria di transito. Senza X$ il contenuto originale di A$ andrebbe perso perchè su di esso si sovrappone il contenuto di B$; infatti:

Istruzione	Memoria		
	A$	B$	
⋮	3	5	
A$=B$	5	5	Errato
B$=A$	5	5	

Più avanti nel corso del programma avremo bisogno di sapere se le due variabili sono state commutate. Per avere questa informazione poniamo un "marker" (o "flag" o segno di avviso) che appunto ci segnali che A$ e B$ sono state scambiate. Se S=0 il cambiamento non è avvenuto, se S=1 il cambiamento c'è stato. Alla linea 90 si pone appunto il marker S:

```
90 S=1
```

Se S=1 è chiaro che il risultato della sottrazione sarà negativo. La risposta negativa

si otterrà concatenando un segno meno davanti a tutto il risultato prima di stamparlo o visualizzarlo. Questo si effettua alla linea 1140:

```
1140 IF S=1 THEN C$="-"+C$
```

Passo 4: Sia che il risultato sia positivo che negativo, abbiamo ora A$ maggiore di B$ per cui possiamo effettuare una normale sottrazione. Tale sottrazione appare alle linee da 1000 a 1080 ed è analoga alle linee da 1020 a 1100 del passo 3 dell'addizione con stringhe numeriche, in quanto le cifre sono estratte dalle stringhe nello stesso modo. Alla linea 1050 la variabile D di riporto è ora usata come riporto a "prestito". Se (A−B) < 0 allora una decina deve essere presa a prestito dalla colonna a sinistra facendo aumentare il valore di A di 10. D è allora posto eguale a −1 perchè "1" è stato imprestato alla colonna di destra. In complesso avremo:

```
1000 REM**SUBTRACTION ROUTINE**
1010 FOR I=LEN(A$) TO 1 STEP-1
1020 A=VAL(MID$(A$,I,1))
1030 A=A+D:D=0
1040 B=VAL(MID$(B$,I,1))
1050 IF (A-B)<0 THEN D=-1:A=A+10
1060 C=A-B
```

										+9		
				+10	+10	+10				+10	+10	
A	1	2	3	3~~4~~	4~~5~~	5~~8~~	7	8	8~~9~~	~~0~~	1	2
−B	0	0	0	0	5	7	9	4	3	5	7	2
C	1	2	3	3	9	8	8	4	5	4	4	0

Passo 5: Concatenamento dei risultati in una unica stringa. Questa parte del programma è presa direttamente dalla linea 1090 dell'addizione con stinghe numeriche eccetto che la variabile N non è usata perchè non vi sarà mai un riporto finale. Nel nostro programma il concatenamento finale è riportato alle lina 1070:

```
1070 C$=RIGHT$(STR$(C),1)+C$
```

Passo 6: La sottrazione può generare degli zeri posti davanti al risultato. Mediante un ciclo FOR NEXT, dalla linea 1090 alla linea 1120, li eliminiamo usando la funzione VAL e una variabile L come contatore:

```
1090 FOR I=1 TO LEN(C$)
1100 IF VAL(MID$(C$,I,1))=0 THEN L=L+1
1110 IF VAL(LEFT$(C$,I))<>0 THEN I=LEN(C$)
1120 NEXT I
```

Il ciclo FOR NEXT ha estremi 1 e la lunghezza della stringa C$. Ad ogni passo ricerca gli zeri o gli spazi vuoti, da sinistra verso destra, mediante la funzione VAL e

incrementando il contatore L (istruzione 1100). Appena incontra il primo carattere non nullo o non vuoto pone la variabile di ciclo eguale al valore massimo così che il ciclo si interrompe subito.

Una volta che il numero di zeri o spazi vuoti è stato determinato, separiamo tali zeri dalla stringa C$ mediante le funzioni RIGHT$ e LEN alla linea 1130. Estrarremo infatti i LEN (C$)−L caratteri più a destra della stringa risultato C$:

```
C$ =[0][0][1][2][3][5][7]                    LEN(C$) = 7
 I      MID$(C$,I,1)
 1       0 0 1 2 3 5 7      = 0              L = 1
 2       0 0 1 2 3 5 7      = 0              L = 2
 3       0 0 1 2 3 5 7    < > 0              I = LEN(C$)
 7                                           I = 7 esce dal ciclo
1130 C$=RIGHT$(C$,LEN(C$)-L)
     C$=RIGHT$ ([0][0][1][2][3][5][7],7-2)
     C$=RIGHT$ ([0][0][1][2][3][5][7],5)
     C$=[1][2][3][5][7]
```

Passo 7: Visualizzazione del risultato C$. Prima di far uscire il risultato finale dobbiamo vedere se esso è negativo controllando il valore del marker S alla linea 1140. Se S=1 allora A$ è minore di B$ e quindi il risultato è negativo e dobbiamo porre il segno meno davanti a C$. Se invece S=0 allora il risultato è positivo e non dobbiamo aggiungere nulla a C$. Alla linea 1150 stampiamo il valore di C$:

```
1140 IF S=1 THEN C$="-"+C$
1150 PRINT:PRINT"ANSWER=";C$:PRINT
```

Alle linee da 1160 a 1180 annulliamo tutte le stringhe e le variabili così da poter rimandare il programma al suo inizio per accettare nuovi dati in ingresso. Il programma completo è il seguente:

```
10 PRINT"□***SUBTRACTION***" PRINT          Pulisce lo schermo
20 INPUT A$,B$                              Ingresso stringhe numeriche
30 BLANK$="                                ")
40 X=LEN(A$) Y=LEN(B$)                      Allineamento a destra
50 IF X<Y THEN A$=LEFT$(BLANK$,Y-X)+A$      (come linee 20-60 del pro-
60 IF Y<X THEN B$=LEFT$(BLANK$,X-Y)+B$      gramma di addizione)

65 IF VAL(A$)=VAL(B$) THEN C$="0":GOTO 1150
70 IF VAL(A$)>=VAL(B$) GOTO 1000            Se A$ < B$, commuta
80 X$=A$:A$=B$:B$=X$                        le stringhe
90 S=1
```

```
1000 REM**SUBTRACTION ROUTINE**
1010 FOR I=LEN(A$) TO 1 STEP-1
1020 A=VAL(MID$(A$,I,1))
1030 A=A+D D=0
1040 B=VAL(MID$(B$,I,1))
1050 IF (A-B)<0 THEN D=-1 A=A+10
1060 C=A-B
1070 C$=RIGHT$(STR$(C),1)+C$
1080 NEXT I
1090 FOR I=1 TO LEN(C$)
1100 IF VAL(MID$(C$,I,1))=0 THEN L=L+1
1110 IF VAL(LEFT$(C$,I))<>0 THEN I=LEN(C$)
1120 NEXT I
1130 C$=RIGHT$(C$,LEN(C$)-L)
1140 IF S=1 THEN C$="-"+C$
1150 PRINT PRINT"ANSWER=";C$ PRINT
1160 C$="" A$="" B$="" X$=""
1165 A=0 B=0 C=0 D=0 S=0 X=0 Y=0
1170 GOTO20
1180 END
***SUBTRACTION***
?123456789012
??57943572
ANSWER= 123398845440
```

Ciclo di sottrazione (analogo alle linee 1020-1100 del programma di addizione)

Troncamento degli zeri anteriori e degli spazi vuoti

Stampa la risposta

Azzeramento

Il programma per la sottrazione di stringhe appena descritto presenta però un problema: se il minuendo e il sottraendo hanno lo stesso numero di cifre e se in ambedue i casi le nove cifre più significative sono eguali, allora il risultato è nullo anche se le altre cifre sono diverse. Per esempio se provate a sottrarre 123456789000 da 123456789012 otterrete 0 pur non essendo i due valori eguali. Questo errore deriva dall'istruzione della linea 65 in cui viene usata la funzione VAL per convertire in numero le due stringhe A$ e B$. La funzione VAL infatti converte solo le nove cifre più significative di sinistra e trascura le altre! Provate voi a proporre una soluzione diversa per risolvere questo problema; per esempio potete confrontare separatamente le parti alte e le parti basse dei due numeri.

Sottrazione di interi multipli

Come nel caso precedente dell'addizione, la tecnica degli interi multipli permette di ripartire un numero in segmenti più piccoli e di calcolare separatamente la sottrazione di segmenti corrispondenti. Il risultato complessivo si ottiene unendo poi i singoli risultati parziali. Ecco i passi della procedura:

1. Caricate il minuendo e il sottraendo come stringhe numeriche positive.
2. Determinate quale stringa ha il valore maggiore.
3. Dividete i numeri nelle parti alte e basse.
4. Calcolate la differenza separatamente per le due parti alte e per le due parti basse.

5. Concatenate i due risultati in un unico risultato.
6. Troncate gli zeri a sinistra.
7. Visualizzate la risposta.

Passo 1: Caricate il minuendo e il sottraendo come due stringhe numeriche positive:

```
10 PRINT"***MULTIPLE INTEGER SUBTRACTION***":PRINT
20 INPUT A$,B$

RUN

***MULTIPLE INTEGER SUBTRACTION***

?123456789012
??57943572
```

Il minuendo A$ e il sottraendo B$ sono caricati come stringhe così da superare la limitazione delle 9 cifre massime.

Come nel caso dell'addizione, poniamo che i due numeri abbiano al massimo 16 cifre così da poterli dividere in due segmenti di 8 cifre ciascuno.

Passo 2: Determinare quale stringa ha il valore maggiore. Se A$ è eguale a B$ il programma salta alla linea 1190 dove stampa il risultato zero. Se B$ è maggiore di A$ il risultato è negativo e le due stringhe sono commutate tra loro così da avere sempre il valore maggiore in A$. Dopo il calcolo della loro differenza si porrà un segno meno davanti alla stringa C$. Come nel caso della sottrazione con stringhe numeriche, anche qui si pone un marker S per ricordare che le due stringhe sono state commutate e il segno del risultato è negativo:

```
30 IF VAL(A$)>VAL(B$) THEN 1000
40 X$=A$:A$=B$:B$=X$
50 S=1
```

Alla linea 50 viene posto il marker e alla linea 40 vengono commutate tra loro A$ e B$ come abbiamo già visto nel paragrafo precedente.

Passo 3: Ripartizione di A$ e B$ in due segmenti più piccoli: le parti alte e le parti basse.

```
1000 X=LEN(A$):Y=LEN(B$)
1002 IF X>Y THEN F=X/2:GOTO 1006
1004 F=Y/2
1006 IF F>INT(F) THEN F=INT(F)+1
1010 IF X<=F THEN AH=0:AL=VAL(A$):GOTO 1040
1020 AH=VAL(LEFT$(A$,X-F))
1030 AL=VAL(RIGHT$(A$,F))
1040 IF Y<=F THEN BH=0:BL=VAL(B$):GOTO 1070
1050 BH=VAL(LEFT$(B$,Y-F))
1060 BL=VAL(RIGHT$(B$,F))
```

Il divisore F è calcolato alle linee da 1002 a 1006. Alle linee 1010 e 1040 le lunghezze delle due stringhe sono confrontate con il divisore F come abbiamo già fatto nel caso analogo dell'addizione. Se una stringa è più corta di F allora la parte alta AH o BH è posto eguale a zero, mentre la parte bassa AL o BL prende tutti i caratteri della stringa. Se invece essa è più lunga di F viene divisa nelle due parti con AH o BH che prende le cifre a sinistra delle otto di destra:

```
A$ 1 2 3 4 5 6 7 8 9 0 1 2
B$         5 7 9 4 3 5 7 2
AH Ø123456                AL Ø789012
BH ØØØØØ57                BL Ø943572
```

Le linee da 1000 a 1060 sono simili alle stesse linee del programma per l'addizione con gli interi multipli. Se avete bisogno di chiarimenti potete vedere il passo 2 che riguarda l'addizione.

Passo 4: Calcolate la differenza separatamente tra le due parti alte e tra le due parti basse. BL è sottratto a AL e BH è sottratto a AH:

```
AH Ø123456                AL Ø789012
BH ØØØØØ57                BL Ø943572
```

Prima però di effettuare le sottrazioni bisogna controllare che il minuendo sia maggiore del sottraendo. Se BL è maggiore di AL allora la differenza è negativa. Questo crea dei problemi perchè CL negativo non può essere concatenato a CH:

```
CH Øxxxxxx + CL  xxxxxx = C Øxxxxxx xxxxxx  Errato
```

Di conseguenza dobbiamo prendere un prestito da AH per avere una differenza positiva tra AL e BL. Con le linee da 1070 a 1090 prendiamo un prestito da AH per incrementare AL:

```
1070 IF AL>=BL THEN 1100
1080 AL=AL+10↑F
1090 AH=AH-1
```

Se AL è maggiore di BL saltate direttamente alla sottrazione. Diversamente dobbiamo prendere un prestito di un milione da AH per aumentare AL:

```
        1  ──────────→  +1000000
AH xxxxxx                AL  xxxxxx
BH xxxxxx                BL  xxxxxx
CH xxxxxx                CL  xxxxxx
```

Una decina è sommata alla cifra più a sinistra di AL. Oppure in maniera più semplice possiamo aggiungere a AL la F-esima potenza di 10:

```
AL=AL+10↑F
```

Nel nostro caso appunto AL è minore di BL come è stato verificato alla linea 1070.

```
AL[ø789012] <BL[ø943572]
```

Di conseguenza dobbiamo prendere in prestito 1000000 (10 ↑ F = 10 ↑ 6 = 1000000) da AH per aumentare AL:

```
1080 AL=AL+10↑F
     AL=AL+10↑6
     AL=AL+1000000
     AL=[789012] + 1000000
     AL=[1789012]
```

Dopo che AL è stato incrementato bisogna diminuire AH di 1:

```
1090 AH=AH-1
     AH=[ø123456]  [ø1]
     AH=[ø123455]
```

A questo punto possiamo effettuare le due sottrazioni e ottenere la parte bassa del risultato CL$ e la parte alta CH$.

Le istruzioni da 1100 a 1102 calcolano CL$:

```
1100 CL$=STR$(INT(AL-BL))
     CL$=STR$(ø1789012-ø943572)
     CL$=STR$(ø845340)
     CL$=ø345540
```

Mediante MID$ alla linea 1101 il carattere più a sinistra che rappresenta il segno viene troncato:

```
1101 CL$=MID$(CL$,2,LEN(CL$)-1)
     CL$=MID$([ø][8][4][5][4][4][0],2,6)
     CL$=        [8][4][5][4][4][0]
```

Se la lunghezza di CL$ è inferiore a F, allora viene riportata a F mediante l'aggiunta di zeri presi dalla stringa ZERO$ della linea 15:

```
15 ZERO$="0000000000000000"
1102 CL$=LEFT$(ZERO$,F-LEN(CL$))+CL$
     CL$=LEFT$ (ZERO$,6-6)+CL$
     CL$=LEFT$ (ZERO$,0)+CL$
```

Alla linea 1110 viene calcolato il valore di CH$:

```
1110 CH$=STR$(INT(AH-BH))
     CH$=STR$(Ø123455-Ø57)
     CH$=STR$(Ø123398)
     CH$=Ø123395
```

Anche in questo caso il segno viene troncato mediante MID$:

```
1111 CH$=MID$(CH$,2,LEN(CH$)-1)
     CH$=MID$(Ø123398,2,6)
     CH$= 123398
```

In conclusione la routine di sottrazione è la seguente:

```
1070 IF AL>=BL GOTO 1100
     789012>=943572 ──────► Istruzione falsa
     Il programma continua alla linea successiva
1080 AL=AL+10↑F
     AL=789012+1000000
     AL=1789012
1090 AH=AH-1
     AH=123456-1
     AHØ123455
1100 CL$=STR$(INT(AL-BL))
     CL$=STR$(Ø1789012-Ø943572)
     CL$=STR$(Ø845540)
     CL$=845540
1101 CL$=MID$(CL$,2,LEN(CL$)-1)
     CL$=MID$(Ø845540,2,7-1)
     CL$=MID$(Ø845540,2,6)
     CL$=845540
1102 CL$=LEFT$(ZERO$,F-LEN(CL$))+CL$
     CL$=LEFT$(ZERO$,6-6)+CL$
     CL$=LEFT$(ZERO$,0)+845540
     CL$=845540
```

```
1110 CH$=STR$(INT(AH-BH))
     CH$=STR$(Ø123455-Ø57)
     CH$=STR$(Ø123398)
     CH$=Ø12339 8
1111 CH$=MID$(CH$,2,LEN(CH$)-1)
     CH$=MID$(Ø12339 8,2,7-1)
     CH$=MID$(Ø12339 8,2,6)
     CH$=12339 8
```

Passo 5: Concatenate le stringhe parziali CH$ e CL$:

Ricordate però che solo gli F numeri più a destra di CL$ saranno concatenati per evitare di includere il segno di CL$.

```
1120 C$=CH$+CL$
     C$=CH$[    ]+ CL$[    ]
     C$= [        ]
```

Passo 6: Troncate gli zeri a sinistra in C$ prima che sia visualizzata. Questo troncamento viene fatto nello stesso modo già visto al passo 5 della sottrazione con stringhe numeriche. Ecco le istruzioni necessarie:

```
1130 FOR I=1 TO LEN(C$)
1140 IF VAL(MID$(C$,I,1))=0 THEN L=L+1
1150 IF VAL(LEFT$(C$,I))<>0 THEN I=LEN(C$)
1160 NEXT I
1170 C$=RIGHT$(C$,LEN(C$)-L)
1180 IF S=1 THEN C$="-"+C$
```

Se il marker S è uguale a 1 allora il risultato è negativo e un segno meno deve essere concatenato a C$ (linea 1180).

Passo 7: Visualizzate la risposta e annullate le variabili e le stringhe prima di ritornare all'inizio della procedura:

```
1190 PRINT PRINT"ANSWER= ",C$ PRINT
1200 A$="" B$="" C$="" CH$="" CL$=""
1205 AH=0 AL=0 BH=0 BL=0 F=0:S=0 X=0 Y=0
1210 GOTO 20
1220 END
```

In definitiva il programma completo è il seguente:

```
10 PRINT"□***MULTIPLE INTEGER SUBTRACTION***":PRINT
15 ZERO$="0000000000000000"
20 INPUT A$,B$
```

```
25 IF VAL(A$)=VAL(B$) THEN C$="0":GOTO 1190
30 IF VAL(A$)>VAL(B$) GOTO 1000
40 X$=A$ A$=B$ B$=X$
50 S=1
1000 X=LEN(A$) Y=LEN(B$)
1002 IF X>Y THEN F=X/2 GOTO 1006
1004 F=Y/2
1006 IF F>INT(F) THEN F=INT(F)+1
1010 IF X<=F THEN AH=0 AL=VAL(A$) GOTO 1040
1020 AH=VAL(LEFT$(A$,X-F))
1030 AL=VAL(RIGHT$(A$,F))
1040 IF Y<=F THEN B=0 BL=VAL(B$) GOTO 1070
1050 BH=VAL(LEFT$(B$,Y-F))
1060 BL=VAL(RIGHT$(B$,F))
1070 IF AL>=BL GOTO 1100
1080 AL=AL+10↑F
1090 AH=AH-1
1100 CL$=STR$(INT(AL-BL))
1101 CL$=MID$(CL$,2,LEN(CL$)-1)
1102 CL$=LEFT$(ZERO$,F-LEN(CL$))+CL$
1110 CH$=STR$(INT(AH-BH))
1111 CH$=MID$(CH$,2,LEN(CH$)-1)
1120 C$=CH$+CL$
1130 FOR I=1 TO LEN(C$)
1140 IF VAL(MID$(C$,I,1))=0 THEN L=L+1
1150 IF VAL(LEFT$(C$,I))<>0 THEN I=LEN(C$)
1160 NEXT I
1170 C$=RIGHT$(C$,LEN(C$)-L)
1180 IF S=1 THEN C$="-"+C$
1190 PRINT PRINT"ANSWER= ";C$ PRINT
1200 A$="" B$="" C$="" CH$="" CL$=""
1205 AH=0 AL=0 BH=0 BL=0 F=0 S=0 X=0 Y=0
1210 GOTO 20
1220 END

***MULTIPLE INTEGER SUBTRACTION***

?123456789012
??57943572

ANSWER= 123398845440

?1234567890123456
??57943572

ANSWER= 1234567832179884

?9999999999999999
??1234567890

ANSWER= 9999998765432109
```

Abbiamo così visto due metodi che ci permettono di effettuare la sottrazione con numeri di lunghezza maggiore di nove cifre.

MOLTIPLICAZIONE

Nel caso della moltiplicazione è molto facile raggiungere il limite delle nove cifre anche quando il moltiplicatore e il moltiplicando sono abbastanza piccoli. Questa limitazione numerica ci obbliga infatti a rappresentare, i numeri maggiori di nove cifre, con la forma esponenziale. **Vedremo adesso che grazie alla tecnica degli interi multipli è possibile rappresentare, non in forma esponenziale, prodoti con più di nove cifre.** Come nei casi precedenti limitiamo la nostra trattazione per semplicità a prodotti fino a 16 cifre.

Moltiplicazione di interi multipli

Come abbiamo già visto per l'addizione e la sottrazione anche nel caso della moltiplicazione la tecnica degli interi multipli prevede che il moltiplicando e il moltiplicatore siano ripartiti in segmenti più piccoli. Tali segmenti sono quindi trattati separatamente e poi uniti nel risultato finale.

I passi della procedura sono i seguenti:

1. Caricate il moltiplicando e il moltiplicatore come due stringhe numeriche positive.
2. Dividete le stringhe nei segmenti alti e bassi.
3. Moltiplicate i segmenti corrispondenti.
4. Addizionate i segmenti prodotto per creare una unica stringa; troncate gli zeri anteriori.
5. Visualizzate il risultato finale.

Passo 1: Caricate il moltiplicando e il moltiplicatore come due stringhe numeriche positive. Poniamo sia A$ il moltiplicando e B$ il moltiplicatore. Anche in questo caso possiamo ovviare alla limitazione delle nove cifre imposta per le grandezze numeriche.

Attenzione però che, nel caso della moltiplicazione, la lunghezza in cifre del prodotto è eguale alla somma delle lunghezze dei due fattori di moltiplicazione. Noi ci limitiamo a 16 cifre per il prodotto per cui le due stringhe A$ e B$ devono avere lunghezze la cui somma non superi 16. Se lo ritenete opportuno provate da voi ad estendere il campo di applicazione di questa routine.

Nel nostro caso dobbiamo quindi avere:

(Lunghezza A$)+(Lunghezza B$) $\leq$ 16

Esempi:	12	+	4	$\leq$16
	2	+	3	$\leq$16
	8	+	8	$\leq$16

Noi stabiliamo di moltiplicare due numeri di otto cifre ciascuno: 99999999 e 99999999 che danno come risultato un numero di 16 cifre:

```
        99999999←        8 cifre
       ×99999999←   +    8 cifre
9999999800000001←       16 cifre
```

Caricate il moltiplicando e il moltiplicatore come due stringhe numeriche positive, A$ e B$:

```
10 PRINT "***MULTIPLE INTEGER MULTIPLICATION***":PRINT
20 INPUT A$,B$
RUN
***MULTIPLE INTEGER MULTIPLICATION***
?99999999
??99999999
```

Passo 2: Dividete le stringhe nei segmenti alti e bassi: AH e BH sono i segmenti alti e AL e BL sono i segmenti bassi. Il fattore di ripartizione è indicato con F. F viene calcolato alle linee da 1002 a 1006 (per le spiegazioni vi rimandiamo all'analogo caso riguardante l'addizione):

```
1000 X=LEN(A$):Y=LEN(B$)
     X=8          Y=8
1002 IF X>Y THEN F=X/2:GOTO 1008
1004 F=Y/2
     F=8/2
     F=4
1006 IF F>INT(F)THEN F=INT(F)+1
```

Appena definito il valore di F il programma ripartisce i due numeri nella parte alta e in quella bassa con la stessa tecnica già vista nei casi precedenti:

```
1010 IF X<=F THEN AH=0:AL=VAL(A$):GOTO 1040
1020 AH=VAL(LEFT$(A$,X-F))
1030 AL=VAL(RIGHT$(A$,F))
1040 IF Y<=F THEN BH=0:BL=VAL(B$):GOTO 1070
1050 BH=VAL(LEFT$(B$,Y-F))
1060 BL=VAL(RIGHT$(B$,F))
```

In questo caso A$ è diviso nelle due parti AH e AL di quattro cifre ciascuna e B$ nelle due parti BH e BL anche loro di quattro cifre:

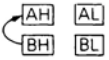
Passo 3: Moltiplicate tra loro AH, AL, BH e BL così da ottenere i quattro prodotti: P1$, P2$, P3$ e P4$. Questi quattro prodotti parziali si ottengono con le stesse regole della moltiplicazione algebrica come se AH, AL, BH e BL fossero delle singole cifre:

Moltiplicare A$ per B$ richiede quattro passi successici. Per prima cosa moltiplicate AL per BL:

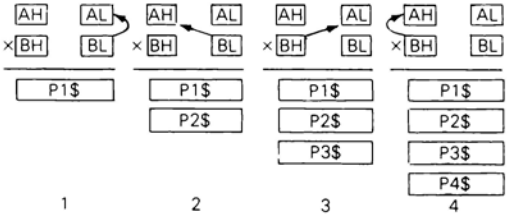
quindi AH per BL:

Successivamente AL per BH:

e infine AH per BH:



Nel complesso il processo completo di moltiplicazione è il seguente:



Vediamo adesso con molta attenzione come si svolgono le singole operazioni, assegnando i loro valori a AH, AL, BH e BL:

La prima moltiplicazione è AL per BL:

```
   AH   [ø9999]
 × BH   [ø9999]
 ------------
        89991
       89991
      8999'1
     89991
    ----------
    99980001
```

La seconda è AH per BL:

```
AH [9999]      [AL]
 ×   [BH]   BL [9999]
 ------------------
            99980001
       999800010000
```

Notate che P2 non è direttamente sottostante a P1, ma è spostato di quattro posti a sinistra (ricordate infatti le regole per incolonnare i prodotti parziali in una moltiplicazione manuale). Il terzo prodotto si otterrà quindi così:

```
   [AH]    AL [9999]
BH [9999]     [BL]
 ----------------
   999800010000
```

Il quarto e ultimo prodotto parziale sarà allora:

```
AH [9999]   [AL]
BH [9999]   [BL]
 ----------------
 9998000100000000
```

Nel nostro esempio però tutti e quattro i segmenti parziali hanno lo stesso contenuto numerico per cui i prodotti sono sempre eguali a 99980001. L'unico aspetto, che distingue un prodotto parziale da un altro, è il diverso numero di zeri che devono essere posti a destra per ottenere poi l'esatto concatenamento. Questo allineamento con gli zeri viene fatto dalla linea 1070 alla linea 1100:

```
1070 P1$=STR$(BL*AL)
1080 P2$=STR$(BL*AH)+F$
1090 P3$=STR$(BH*AL)+F$
1100 P4$=STR$(BH*AH)+F$+F$
```

Se non si fossero aggiunti gli zeri avremmo avuto questo concatenamento inesatto:

```
P1  99980001
P2  99980001   Errato
P3  99980001
P4  99980001
```

invece di quello corretto:

```
          99980001
     99980001 0000
     99980001 0000   Corretto
 99980001 00000000
```

Gli zeri vengono aggiunti mediante la stringa F$ che viene creata con queste istruzioni:

```
40 ZERO$="0000000000000000"
1008 F$=LEFT$(ZERO$,F)
     F$=LEFT$(ZERO$,4)
     F$=0000000000
     F$="0000"
```

Con le istruzioni dalla linea 1070 alla linea 1100 i prodotti parziali P1$, P2$, P3$ e P4$ vengono calcolati e allineati in modo da poter essere subito sommati:

```
   AH 9999    AL 9999
 × BH 9999    BL 9999
 ------------------------
            99980001  P1
        999800010000  P2
        999800010000  P3
    9998000100000000  P4
            F$   F$
```

Al termine del passo 3 il programma è il seguente:

```
20 INPUT A$,B$                                  Ingresso dei valori di A$ e B$
30 IF VAL(A$)=0 OR VAL(B$)=0 THEN              } Se il moltiplicando o il mol-
   C$="0":GOTO 1190                            } tiplicatore = 0 allora la ri-
40 ZERO$="0000000000000000"                    } sposta (C$) = 0
1000 X=LEN(A$):Y=LEN(B$)                       }
1002 IF X>Y THEN F=X/2:GOTO 1006               } calcolo del fattore F
1004 F=Y/2                                     }
1006 IF F>INT(F)THEN F=INT(F)+1                }
```

```
1008 F$=LEFT$(ZERO$,F)
1010 IF X<=F THEN AH=0:AL=VAL(A$):GOTO 1040
1020 AH=VAL(LEFT$(A$,X-F))
1030 AL=VAL(RIGHT$(A$,F))
1040 IF Y<=F THEN BH=0:BL=VAL(B$):GOTO 1070
1050 BH=VAL(LEFT$(B$,Y-F))
1060 BL=VAL(RIGHT$(B$,F))
1070 P1$=STR$(BL*AL)
1080 P2$=STR$(BL*AH)+F$
1090 P3$=STR$(BH*AL)+F$
1100 P4$=STR$(BH*AH)+F$+F$
```

Ripartizione di A$ e B$ nelle parti alte e basse

Moltiplicazione di A$ e B$ e allineamento dei prodotti parziali

Passo 4: Addizione dei quattro prodotti parziali. Questa è la parte più complicata, del metodo di moltiplicazione per interi multipli, in quanto si devono trasferire e ricevere parametri tra il programma principale e una subroutine. Useremo infatti una parte del programma di addizione con stringhe numeriche come subroutine per addizionare i prodotti parziali:

```
2000 REM**ADD PRODUCTS**
2010 BLANK$="                    "
2020 X=LEN(A$):Y=LEN(B$)
2030 IF X<Y THEN A$=LEFT$(BLANK$,Y-X)+A$
2040 IF X>Y THEN B$=LEFT$(BLANK$,X-Y)+B$
2050 D=0:N=1:C$=""
2060 FOR I=LEN(A$) TO 1 STEP-1
2070 A=VAL(MID$(A$,I,1))
2080 A=A+D:D=0
2090 B=VAL(MID$(B$,I,1))
2100 C=A+B
2110 IF C>=10 THEN D=1
2120 IF D=1 AND I=1 THEN N=2
2130 C$=RIGHT$(STR$(C),N)+C$
2140 NEXT I
```

Alla linea 1110 i contenuti di P1$ e P2$ sono trasferiti come parametri A$ e B$ che saranno poi usati nella subroutine (dalla linea 2000 alla linea 2140):

```
1110 A$=P1$:B$=P2$
     A$  9998000 1
     B$  9998000 10000
```

Notate che le variabili A$ e B$ non sono le stesse della linea 20. Si sono usati gli stessi nomi solo per uniformità tra tutte le routine matematiche che vi abbiamo presentato in questo capitolo. Attenzione poi che alla subroutine 2000 sono stati trasferiti solo due dei quattro prodotti parziali.

Dopo aver trasferito i valori di P1$ e P2$ a A$ e B$ bisogna chiamare la subroutine di addizione:

```
1120 GOSUB 2000
```

A$ e B$ sono allineati a destra e posti di eguale lunghezza alle linee da 2010 a 2040:

```
2010 BLANK$="                    "
2020 X=LEN(A$):Y=LEN(B$)
2030 IF X<Y THEN A$=LEFT$(BLANK$,Y-X)+A$
2040 IF X>Y THEN B$=LEFT$(BLANK$,X-Y)+B$
```

Ricordiamo che questa subroutine impiega lo stesso metodo del programma per l'addizione di due numeri con la tecnica delle stringhe numeriche. Le istruzioni dalla linea 2050 alla linea 2140 determinano il valore della stringa C$ come somma delle cifre corrispondenti di A$ e B$:

```
2050 D=0:N=1:C$=""
2060 FOR I=LEN(A$) TO 1 STEP-1
2070 A=VAL(MID$(A$,I,1))
2080 A=A+D:D=0
2090 B=VAL(MID$(B$,I,1))
2100 C=A+B
2110 IF C>=10 THEN D=1
2120 IF D=1 AND I=1 THEN N=2
2130 C$=RIGHT$(STR$(C),N)+C$
2140 NEXT I
```

Una volta ottenuta C$ bisogna troncarle gli eventuali zeri posti anteriormente. Questo viene fatto con un programma analogo a quello per la sottrazione con stringhe numeriche:

```
3000 REM***TRUNCATE LEAD ZEROS***
3001 L=0
3010 FOR I=1 TO LEN(C$)
3020 IF VAL(MID$(C$,I,1))=0 THEN L=L+1
3030 IF VAL(LEFT$(C$,I))<>0 THEN I=LEN(C$)
3040 NEXT I
3050 C$=RIGHT$(C$,LEN(C$)-L)
3060 RETURN
```

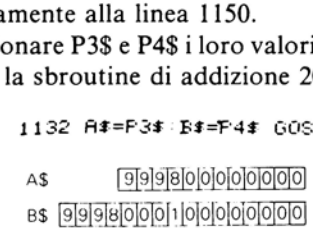
Il valore di C$ viene quindi ritornato al programma principale e posto eguale a M1$:

```
1130 M1$=C$
```

Ovviamente C$ deve essere trasferito in un'altra variabile M1$ perchè sarà utilizzato successivamente alla linea 1150.

Per addizionare P3$ e P4$ i loro valori sono trasferiti ai parametri A$ e B$ prima di chiamare la sbroutine di addizione 2000:

```
1132 A$=P3$:B$=P4$ GOSUB 2000
```



Anche in questo caso la subroutine somma i due valori P3$ e P4$, tronca gli zeri posti a sinistra e restituisce il risultato C$ che sarà posto in M2$:

```
1135 M2$=C$
```

La subroutine di addizione 2000 viene infine chiamata una terza volta per sommare tra loro M1$ e M2$ e dare quindi il risultato finale C$:

```
1140 A$=M1$ B$=M2$
     A$      9999899999000001
     B$ 99998999900000000000
1150 GOSUB 2000
```

Passo 5: A questo punto C$ contiene il risultato finale per cui può essere stampata. Con la istruzione GOTO 20 il programma ritorna all'inizio per effettuare una nuova operazione:

```
1190 PRINT PRINT"ANSWER=",C$ PRINT GOTO 20
1200 END
```

Ricapitoliamo schematicamente lo sviluppo di questo programma per la moltiplicazione di due numeri:

```
         10   PRINT "*** MULTIPLE INTEGER MULTIPLICATION ***"
         20
Passo 1  30   Ingresso moltiplicatore e moltiplicando
         40   e inizializzazione delle variabili
         50
         1000
         1010
         1020 Calcolo di F e ripartizione del
Passo 2  1030 moltiplicatore e del moltiplicando
         1040 nelle parti alte e basse
         1060
         1070
         1080 Calcolo dei quattro prodotti parziali
Passo 3  1090 P1$, P2$, P3$ e P4$
         1100
         1110 P1$ e P2$ passano a A$ e B$
         1120 GOSUB 2000 ---------------------> subroutine di addizione (2000-2140)
                                                P1$ + P2$ → c$
                         <---------------------- (3000–3600) troncamento degli zeri
         1130 C$ passa a M1$
              P3$ e P4$ passano a A$ e B$
Passo 4  1132 GOSUB 2000 ---------------------> subroutine di addizione (2000-2140)
                                                P3$ + P4$ → C$
                         <---------------------- (3000–3600) troncamento degli zeri
              C$ passa a M2$
              M1$ e M2$ passano a A$ e B$
         1150 GOSUB 2000 ---------------------> subroutine di addizione (2000-2140)
                                                M1$ + M2$ → C$
                         <---------------------- (3000–3600) troncamento degli zeri
Passo 5  1190 PRINT C$
         1200 END
```

Il suo listato completo è il seguente:

```
10 PRINT"□***MULTIPLE INTEGER MULTIPLICATION***":PRINT
20 INPUT A$,B$
30 IF VAL(A$)=0 OR VAL(B$)=0 THEN C$="0":GOTO 1190
40 ZERO$="0000000000000000"
1000 X=LEN(A$):Y=LEN(B$)
1002 IF X>Y THEN F=X/2:GOTO 1008
1004 F=Y/2
1006 IF F>INT(F)THEN F=INT(F)+1
1008 F$=LEFT$(ZERO$,F)
1010 IF X<=F THEN AH=0:AL=VAL(A$):GOTO 1040
1020 AH=VAL(LEFT$(A$,X-F))
1030 AL=VAL(RIGHT$(A$,F))
1040 IF Y<=F THEN BH=0:BL=VAL(B$):GOTO 1070
1050 BH=VAL(LEFT$(B$,Y-F))
1060 BL=VAL(RIGHT$(B$,F))
1070 P1$=STR$(BL*AL)
1080 P2$=STR$(BL*AH)+F$
1090 P3$=STR$(BH*AL)+F$
1100 P4$=STR$(BH*AH)+F$+F$
1110 A$=P1$:B$=P2$
1120 GOSUB 2000
1130 M1$=C$
1132 A$=P3$:B$=P4$:GOSUB 2000
1135 M2$=C$
1140 A$=M1$:B$=M2$
1150 GOSUB 2000
1190 PRINT:PRINT"ANSWER=";C$:PRINT:GOTO 20
1200 END
2000 REM**ADD PRODUCTS**
2010 BLANK$="                    "
2020 X=LEN(A$):Y=LEN(B$)
2030 IF X<Y THEN A$=LEFT$(BLANK$,Y-X)+A$
2040 IF X>Y THEN B$=LEFT$(BLANK$,X-Y)+B$
2050 D=0:N=1:C$=""
2060 FOR I=LEN(A$) TO 1 STEP-1
2070 A=VAL(MID$(A$,I,1))
2080 A=A+D:D=0
2090 B=VAL(MID$(B$,I,1))
2100 C=A+B
2110 IF C>=10 THEN D=1
2120 IF D=1 AND I=1 THEN N=2
2130 C$=RIGHT$(STR$(C),N)+C$
2140 NEXT I
3000 REM***TRUNCATE LEAD ZEROS***
3001 L=0
3010 FOR I=1 TO LEN(C$)
3020 IF VAL(MID$(C$,I,1))=0 THEN L=L+1
3030 IF VAL(LEFT$(C$,I))<>0 THEN I=LEN(C$)
3040 NEXT I
3050 C$=RIGHT$(C$,LEN(C$)-L)
3060 RETURN
```

```
*** MULTIPLE INTEGER MULTIPLICATION***

?99999999
??999999999

ANSWER= 9999999800000001
```

GRAFICA

La rappresentazione grafica con i calcolatori è un argomento di notevole interesse su cui sono stati scritti molti libri. In questo capitolo ci limitiamo a riportare le possibilità di programmazione grafica più importanti ottenibili con i calcolatori CBM.

L'insieme di caratteri grafici standard comprende 64 simboli. Se state usando l'insieme di caratteri alternativo date il comando POKE 59468, 12 per ottenere i caratteri grafici. **Se avete un calcolatore CBM 8000 potete richiamare i caratteri grafici mediante la funzione di editing grafico in questo modo:**

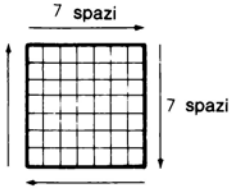
```
100 print chr$(142)
```

I caratteri grafici sono sempre posti sulla parte alta dei tasti e quindi devono essere battuti in modo shiftato.

Molti caratteri grafici sono descritti qui di seguito, ma per un elenco completo vi rimandiamo alla Tabella 1-1 o all'Appendice A.

GRAFICA IN MODO IMMEDIATO

Tracciare un grafico in modo immediato non richiede alcuna istruzione PRINT, alcun numero di linea ne l'uso di virgolette. Potete muovere il cursore verso destra come verso sinistra o in alto e in basso senza mai premere RETURN. Per farvi capire come si "disegna" con il calcolatore vi mostriamo come tracciare questo quadrato:



Anche se può sembrare molto semplice disegnare un quadrato in realtà ciò presenta tutti i problemi caratteristici della grafica con calcolatore.

Disegno di un quadrato

Sono necessari nove passi per disegnare un quadrato 7 x 7.
Essi sono:

Passo 1: Portate in HOME il cursore. La posizione in alto a sinistra di HOME è presa come angolo di sinistra alto del quadrato (Figura 5-3a).

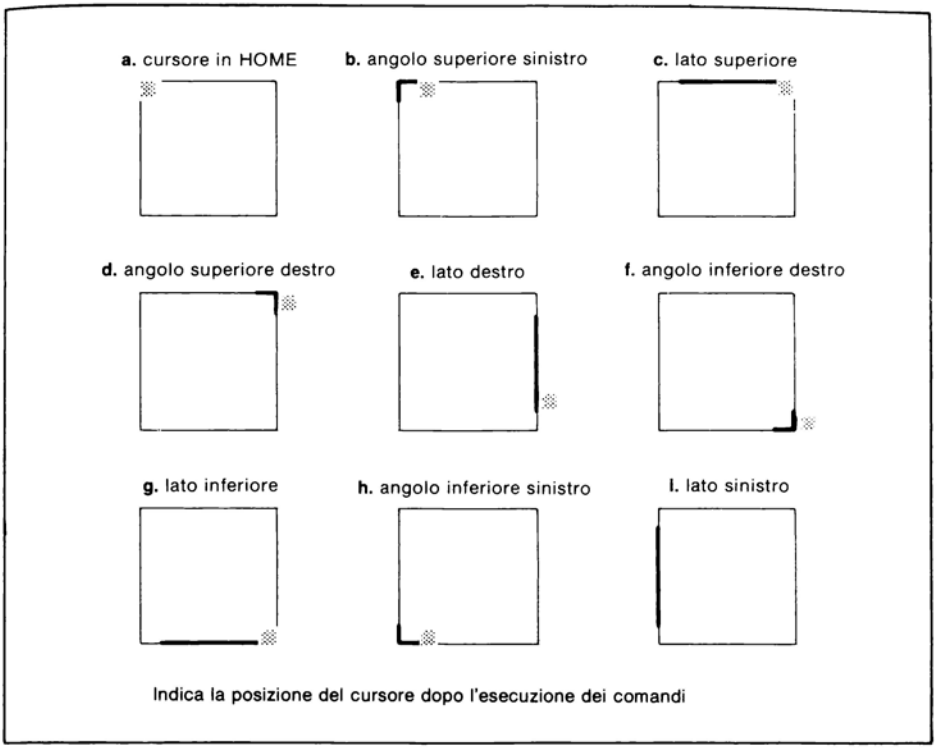


Figura 5-3: Disegno di un quadrato

Passo 2: Battete l'angolo di sinistra in alto del quadrato mediante il tasto TOP LEFT CORNER (Figura 5-3b).

Passo 3: Tracciate il lato superiore mediante il tasto TOP LINE HORIZONTAL battuto cinque volte (Figura 5-3c).

Passo 4: Tracciate l'angolo superiore destro mediante il tasto TOP RIGHT CORNER (Figura 5-3d).

Passo 5: Tracciate il lato verticale destro battendo cinque volte RIGHT LINE VERTICAL.

È molto facile che a questo punto invece di tracciare una linea verticale ottenere qualcosa del genere:

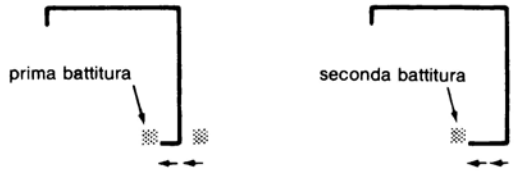
Questo è avvenuto perchè il cursore si spostava automaticamente ad ogni battuta a destra. Per tracciare i caratteri verticalmente dovete ad ogni passo riposizionare il

cursore spostandolo in basso con CURSOR DOWN e a sinistra con CURSOR LEFT. In definitiva, per tracciare il lato veriticale destro, dovete battere cinque volte i due comandi del cursore e RIGHT LINE VERTICAL (Figura 5-3e).

Passo 6: Tracciate l'angolo inferiore destro. Anche in questo caso prima di battere il carattere grafico dovete posizionare correttamente il cursore con CURSOR DOWN e CURSOR LEFT. Battete quindi BOTTOM RIGHT CURNER (Figura 5-3f).

Passo 7: Tracciate il lato inferiore. Prima di battere cinque volte BOTTOM LINE HORIZONTAL (Figura 5-3g) dovete pensare di sistemare il cursore.

Un metodo è quello di disegnare la linea da destra verso sinistra per cui dovete ogni volta dare due comandi di CURSOR LEFT:



Un altro metodo è quello di tracciarla da sinistra verso destra portando inizialmente il cursore tutto a sinistra e poi tracciare semplicemente il lato. In questo caso date sei comandi CURSOR LEFT:

Passo 8: Tracciate l'angolo inferiore sinistro. A seconda del metodo scelto per tracciare il lato inferiore dovrete dare due (metodo 1) oppure sei (metodo 2) volte il comando CURSOR LEFT per portare il cursore nell'angolo di sinistra in basso. A questo punto battete BOTTOM LEFT CORNER (Figura 5-3h).

Passo 9: Disegnate il quarto lato verticale destro. Anche in questo caso dovete posizionare il cursore con CURSOR LEFT e CURSOR UP prima di battere LEFT LINE VERTICAL (Figura 5-3i).

GRAFICA A PROGRAMMA

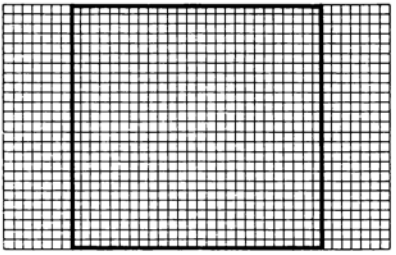
Qualunque cosa abbiate disegnato sullo schermo andrà persa se date il comando NEW oppure se spegnete il calcolatore. Per memorizzare un grafico dovete inserir-

lo in un programma che sarà poi conservato su una cassetta o un dischetto. **Per fare questo basta che ogni riga del disegno sia convertita in una stringa che viene poi inserita in una PRINT di un normale programma.**

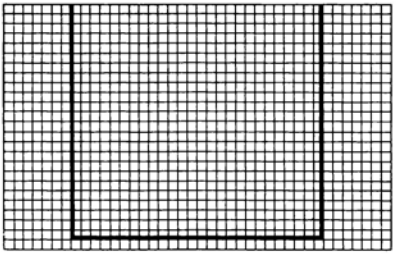
Ritorniamo all'esempio del quadrato. Portiamo il cursore nella posizione HOME facendo molta attenzione a non premere CLEAR o RETURN. Se premete il primo cancellerete tutto lo schermo, se premete il secondo vi apparirà "READY" nel mezzo del quadrato:

Se avete disegnato il quadrato così grande, da avere il lato superiore sulla prima riga e il lato inferiore sull'ultima riga dello schermo, e se il cursore è posizionato su questa riga in basso, allora premendo RETURN avrete uno scorrimento del quadrato verso l'alto di una riga per far posto in basso a "READY":

Prima di RETURN



Dopo di RETURN



Per questa ragione non si devono mai disegnare in modo immediato grafici di dimensioni superiori a 39 x 24 caratteri.

Una volta che il cursore è nella posizione HOME si deve spostare tutto il disegno di alcuni caratteri a destra per far posto ai numeri di linea, al comando PRINT (o ancora meglio ?) e alle virgolette per far sì che tutto il disegno diventi un "programma".

Una volta che avete portato il cursore nell'angolo di sinistra in alto del quadrato (Figura 5-4a), premete INSERT cinque volte così da spostare a destra l'angolo del quadrato (Figura 5-4b). Ora abbiamo abbastanza spazio per battere un numero di linea (100), il comando ? e le virgolette di apertura di una stringa (Figura 5-4c). Premiamo quindi RETURN e così il lato del quadrato diventa una stringa posta in un programma (Figura 5-4d). Nello stesso modo procediamo per tutte le righe che compongono il quadrato incrementando ogni volta il numero di linea (Figura 5-4e, f).

Attenzione a porre i numeri di linea in ordine crescente per non distorcere il quadrato. Notate poi che non è necessario porre le virgolette di chiusura della stringa basta che battiate RETURN appena dopo che avete posto le prime virgolette. Se fate stampare il listato otterrete infine:

```
100 PRINT"┌──────┐
200 PRINT"│      │
300 PRINT"│      │
400 PRINT"│      │
500 PRINT"│      │
600 PRINT"│      │
700 PRINT"└──────┘
```

Nell'esempio precedente abbiamo tracciato il quadrato in modo immediato e poi lo abbiamo inserito in un programma. Volendo si può tracciare un grafico direttamente nell'ambito di un programma purchè ogni sua riga faccia parte di una istruzione PRINT:

```
100 ?"┌──────┐
200 ?"│      │
300 ?"│      │
```

La prima posizione immediatamente a destra delle virgolette diviene quindi la prima colonna utile per tracciare il disegno: attenzione a non dimenticarvene, perchè la parte destra del disegno potrebbe essere ribaltata sulla sinistra dello schermo!

Se scrivete una stringa nella lista della PRINT lunga esattamente 40 caratteri, dovete allora porre le virgolette di chiusura e un punto e virgola in coda all'istruzione. Se non ponete il punto e virgola verrà visualizzata una riga in più in quanto il cursore si posiziona automaticamente a capo dopo aver visualizzato qualcosa alla 40-esima colonna.

Vi diamo un consiglio: disegnate su un pezzo di carta quadrettata il vostro grafico prima di tracciarlo sullo schermo. Se volete inserirlo come programma ricordatevi di tener conto anche dello spazio per i numeri di linea e per le istruzioni.

ANIMAZIONE

Ogni grafico, numero o parola possono essere programmati per muoversi in alto, in basso, in diagonale, fuori dallo schermo oppure possono lampeggiare o muoversi più o meno lentamente. Tali possibilità possono essere usate in qualunque combinazione.

Per illustrarvi come si possa ottenere l'animazione di un grafico cominciamo con l'animare il quadrato dell'esempio precedente. Invece di farlo apparire tutto assieme faremo in modo che esso si formi sullo schermo elemento per elemento.

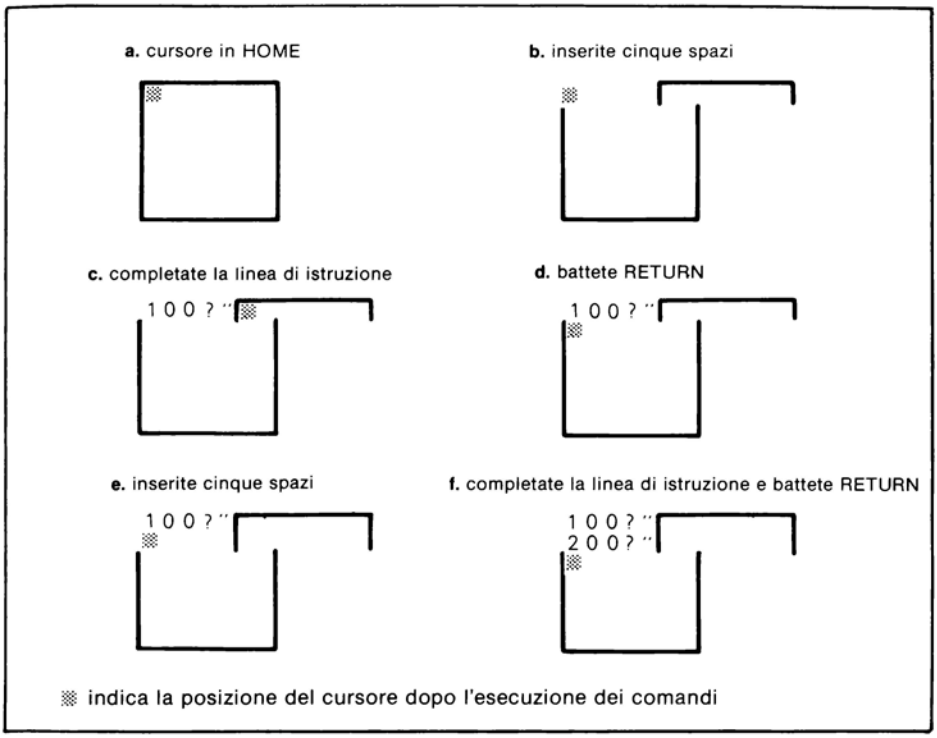


Figura 5-4: Come convertire un grafico in istruzioni di un programma

In questo caso il programma si presenta molto diverso perchè non vi è più il quadrato già formato come tante stringhe ma vi sono solo i caratteri grafici che lo tracceranno durante l'esecuzione del programma.

Ritardo di tempo

Il programma di animazione muove lentamente il cursore così da far sembrare che qualcuno stia disegnando il quadrato. Il tracciato inizia dall'angolo in alto a sinistra e procede in verso orario come indichiamo in questo disegno:

Come sempre il primo passo consiste nel pulire lo schermo. Così facendo si ottiene di portare in "home" il cursore:

```
5 PRINT"□";
```

La seconda istruzione prevede che si tracci l'angolo di sinistra in alto, ma non tutto il lato come si era fatto precedentemente:

```
10 PRINT"┌";
```

A questo punto, prima di andare avanti, dobbiamo pensare di rallentare l'esecuzione delle singole istruzioni perchè con la velocità caratteristica di un calcolatore il nostro quadrato sarebbe tracciato in un tempo brevissimo. Questo rallentmento può essere ottenuto con l'inserimento di una routine di ritardo come la seguente:

```
100 FOR J=1 TO 100:NEXT J:RETURN
```

Il ciclo FOR NEXT inserisce un tempo di attesa tra due caratteri grafici in quanto il calcolatore deve contare da 1 a 100. È chiaro che aumentando o diminuendo l'estremo superiore del ciclo si possono avere tempi diversi di attesa.

Nel nostro programma per semplicità abbiamo posto il ciclo di attesa come se fosse una subroutine con il numero 100, equivalente a circa 1/10 di secondo.

Programmazione del posizionamento dei caratteri grafici

Per tracciare il lato superiore usiamo un ciclo FOR NEXT:

```
15 FOR I=1 TO 5 PRINT"─"; GOSUB 100 NEXT I
```

Come si vede ad ogni esecuzione del ciclo viene richiamata la subroutine 100 di ritardo prima di tracciare il successivo carattere "—". Così facendo il quadrato viene disegnato lentamente. Se invece avessimo posto:

```
15 PRINT"────"; GOSUB 100 ←— Errato
```

tutto il lato superiore sarebbe stato tracciato quasi istantaneamente.

Per completare il lato superiore, scrivete l'istruzione per tracciare l'angolo superiore destro. In coda ponete poi il richiamo alla subroutine di ritardo 100:

```
20 PRINT"┐"; GOSUB 100
```

Fino a questo punto il programma è il seguente:

```
5 PRINT"□";
10 PRINT"┌";:GOSUB 100
15 FOR I=1 TO 5:PRINT"─";:GOSUB 100:NEXT I
20 PRINT"┐";:GOSUB 100
30 END
100 FOR I=1 TO 100:NEXT J:RETURN
```

Ponetelo in esecuzione: vedrete sullo schermo formarsi lentamente il lato superiore da ┌ a ┌──────┐

Se questo non avviene sarà sicuramente dovuto a qualche errore fatto nelle prime righe del programma.

In questo programma ogni PRINT deve essere terminata con un punto e virgola (;) in modo da concatenare assieme le stringhe grafiche. Otterremo così che l'angolo "┌" venga unito al lato "──────" Se mancassero i punti e virgola il calcolatore andrebbe a capo dopo ogni PRINT per cui avremmo sullo schermo qualcosa del genere:

```
┌
└
─
─
─
─
─
┐
```

Gli altri tre lati del quadrato vengono tracciati nello stesso modo. Alla linea 20 inizia la sequenza per tracciare il lato destro. Notate che vengono dati i comandi al cursore per compensare gli spostamenti automatici a destra. Le istruzioni PRINT, per generare i tre lati rimanenti, sono poste ognuna in un ciclo e sono le seguenti:

```
                                PRINT<RIGHT LINE VERT ><CURSOR L >
PRINT" ▮▯" Lato destro          <CURSOR DOWN>
                                PRINT<BOTTOM LINE HORIZ ><CURSOR L >
PRINT"_▮▮" Lato inferiore       <CURSOR L.>
                                PRINT<LEFT LINE VERT ><CURSOR L >
PRINT"I ▮▯" Lato sinistro       <CURSOR UP>
```

Il listato completo del programma è il seguente:

```
5 PRINT"▯";
10 PRINT"┌"; GOSUB 100
15 FOR I=1TO 5 PRINT"─"; GOSUB 100:NEXT I
20 PRINT"┐"; GOSUB 100
25 FOR I=1TO 5 PRINT" ▮▯"; :GOSUB 100:NEXT I
30 PRINT"_"; GOSUB 100
35 FOR I=1TO 5 PRINT"_▮▮"; GOSUB 100:NEXT I
40 PRINT"L"; GOSUB 100
45 FOR I=1TO 5 PRINT"I ▮▯"; :GOSUB 100:NEXT I
50 END
100 FOR J=1 TO 10:NEXT J RETURN
```

Provate ora a porlo in esecuzione. Ottenete per caso qualcosa del genere?

```
┌──────┐I
READY.  I
    I   I
    I   I
    I   I
  L I ────
```

Se succede questo vuol dire che avete dimenticato di posizionare correttamente il cursore. Se analizziamo attentamente il programma vediamo che alle linee 20, 30 e 40 non abbiamo posto il controllo del cursore dopo il tracciamento dell'angolo. Il programma corretto risulta quindi il seguente:

```
5 PRINT"";
10 PRINT"";:GOSUB 100
15 FOR I=1 TO 5:PRINT"";:GOSUB 100:NEXT I
20 PRINT"";:GOSUB 100
25 FOR I=1 TO 5:PRINT" ";:GOSUB 100:NEXT I
30 PRINT"";:GOSUB 100
35 FOR I=1 TO 5:PRINT"";:GOSUB 100:NEXT I
40 PRINT"";:GOSUB 100
45 FOR I=1 TO 5:PRINT"";:GOSUB 100:NEXT I
50 END
100 FOR I=1 TO 100:NEXT J:RETURN
```

Provate ancora ad eseguire il programma. Otterrete il segnale READY sovrapposto al quadrato:

Il quadrato è stato tracciato lentamente come volevamo, ma il segnale READY è apparso sulla seconda riga dello schermo. Per evitare questo è sufficiente porre alcuni comandi di CURSOR DOWN prima di terminare il programma alla linea 50:

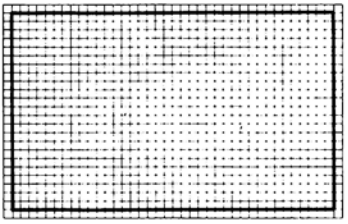
```
50 PRINT"":END
```

Proviamo allora una ultima esecuzione del programma e constatiamo che il quadrato viene visualizzato correttamente:

Allargamento del quadrato

A questo punto vi proponiamo di allargare il quadrato così da formare una cornice dello schermo. Supponiamo sempre che lo schermo sia a 40 colonne per cui vogliamo tracciare un quadrato o meglio dire un rettangolo, con i lati a un

quadratino di distanza dal bordo dello schermo:



Siccome lo schermo ha le dimensioni 40x25, il rettangolo sarà invece 38x23:

40–2 (1 spazio per ogni lato) = 38
25–2 (1 spazio per ogni lato) = 23

Il programma per traciare questo rettangolo può essere ottenuto da quello precedente con poche modifiche. È sufficiente estendere gli estremi superiori dei cicli FOR NEXT che disegnano i lati e portarli ai valori 36 per quelli orizzontali e 21 per quelli verticali. La parte che riguarda gli angoli rimane inalterata:

```
15 FOR I=1 TO 36:?"▔";
25 FOR I=1 TO 21:?" ▮▮";
35 FOR I=1 TO 36:?"▁▁";
45 FOR I=1 TO 21:?"▮▮";
```

Lati orizzontali

Lati verticali

Prima di considerare terminato questo programma dovete decidere dove far apparire il segnale di READY. Se infatti lo lasciate dove era nel programma precedente esso farà scorrere verso l'alto, fuori dello schermo, il rettangolo. L'unica soluzione, ma anche la più interessante è che portiate READY all'interno perchè solo così ha significato aver tracciato una cornice dello schermo.

OROLOGIO IN TEMPO REALE

Un'altra caratteristica dei calcolatori CBM è quella di avere un orologio in tempo reale. Tale orologio dà il tempo in ore, minuti e secondi tramite le stringhe TIME$ o TI$.

Regolazione dell'orogologio

Per regolare l'orologio e in particolare per porre l'ora giusta quando si accende il calcolatore , dovete usare questo formato:

TIME$ = "hhmmss"

dove:

hh ore comprese tra 00 e 23
mm minuti compresi tra 00 e 59
ss secondi compresi tra 00 e 59

Con hh dovete dare le ore comprese tra 00 e 23; a mezzanotte le ore passano da 23 59 59 a 00 00 00.

Quando dovete dare l'ora all'accensione del vostro calcolatore, ponete la stringa TIME$ con qualche secondo in più e poi appena raggiungete quel valore premete RETURN:

```
TIME$="120150"
```

Lettura dell'ora

Per leggere l'ora basta dare il seguente comando:

```
?TIME$
```

e il calcolatore vi risponderà:

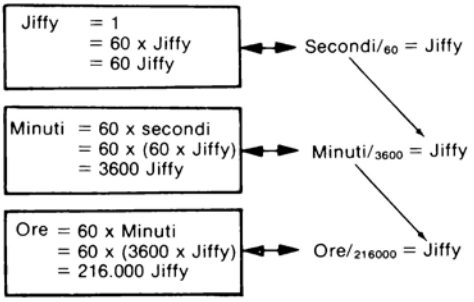
```
?TIME$
120200
```

L'orologio dei calcolatori CBM si mantiene attivo sino a che il calcolatore rimane acceso. **Se spegnete il calcolatore dovete ripetere l'operazione di regolazione vista più sopra al momento della sua riaccensione.**

Uso dell'orologio in tempo reale.

Se volessimo analizzare attentamente il funzionamento dell'orologio vedremmo che **esso scatta ogni 1/60 di secondo. Questo tempo base lo chiameremo "attimo" proprio per la sua brevità (in inglese "jiffy").** Ogni 1/60 di secondo quindi la variabile numerica TIME o TI viene incrementata: essa parte da zero quando il calcolatore viene acceso e va avanti sino a 51.839.999 "jiffy". **La variabile di stringa TIME$ da invece il tempo in ore-minuti-secondi convertendo il tempo di TIME che è dato in "attimi".** Notate che la stringa TIME$ non è semplicemente la conversione in stringa della variabile numerica TIME, ma è la vera decodifica del tempo da "attimi" in ore-minuti-secondi.

Questa conversione è fatta nel modo seguente. Ogni 60 "attimi" costituiscono un secondo; ogni minuto contiene 60 secondi e quindi 3600 "attimi"; ogni ora infine contiene 60 minuti e 216.000 "attimi". Ecco uno schema riepilogativo:



Qui di seguito vi diamo una routine completa per convertire il tempo da "attimi" J in ore H, minuti M e secondi S:

```
10 J=TI
20 H=INT(J/216000)
```

Calcola le ore.
Le funzione INT prende solo la parte intera

```
30 IF H<>0 THEN J=J-H*21600
```

Se vi sono ore, si sottraggono gli "attimi" di ogni ora

```
40 M=INT(J/3600)
```

Calcola i minuti

```
50 IF M<>0 THEN J=J-M*3600
```

Si sottraggono gli "attimi" di ogni minuto

```
60 S=INT(J/60)
```

Calcola i secondi

```
5 PRINT"REAL TIME":PRINT:PRINT:
10 J=TI
15 T$=TIME$
20 H=INT(J/21600)
30 IF H<>0 THEN J=J-H*21600
40 M=INT(J/3600)
50 IF M<>0 THEN J=J-M*3600
60 S=INT(J/60)
70 H$=RIGHT$(STR$(H),2)
80 M$=RIGHT$(STR$(M),2)
90 S$=RIGHT$(STR$(S),2)
100 PRINT"H M S  ";H$;" ";M$;" ";S$,"TIME$: ";T$
110 PRINT"[illegible]";:GOTO10
```

In questo programma le istruzioni alle linee 70, 80 e 90 convertono le variabili dell'ora da numeriche in stringhe. Alla linea 100 vengono stampati sia il tempo calcolato dal programma che quello dato dalla funzione TIME$. Notate che i due valori sono eguali.

Per avere una idea della rapidità dello scorrere del tempo misurato in "attimi" e della conversione degli "attimi" in ora normale, vi proponiamo questo programmino che visualizza sia TIME$ che TIME:

```
5 REM **RUNNING CLOCKS**
10 PRINT "REAL TIME: ":PRINT:PRINT "JIFFY TIME: "
20 FOR I=1 TO 235959
30 PRINT"[illegible]";TAB(13);TIME$
40 FOR J=1 TO 60 STEP 2
50 PRINT"[illegible]";TAB(12);TI
60 NEXT J
70 NEXT I
```

Il ciclo FOR NEXT per la variabile TIME alla linea 40 ha il passo di incremento STEP eguale a 2 (cioè ogni 2 "attimi") per questi due motivi:

1. Visualizzare 60 "attimi" al secondo è troppo veloce da leggere.
2. Visualizzare ogni "attimo" richiede molto più tempo che incrementarli. Questo fa rallentare il ciclo così che la visualizzazione di TIME$ diviene molto più lenta di quello che dovrebbe essere. Eseguite questo programma e vedrete che gli "attimi" sono incrementati di 60 per ogni secondo. Se invece togliete lo STEP 2 alla linea 40 noterete un certo ritardo di tempo nella visualizzazione di TIME$.

```
Tempo reale:          000705
Tempo in "attimi":    25500
```

L'uso del tempo misurato in "attimi" è molto utile quando si debba misurare la velocità di esecuzione di un programma. Queste misure di tempo permettono di valutare l'efficienza di un programma. Consideriamo per esempio questo piccolo programma:

```
10 PRINT"⊃**KEYBOARD TEST**":PRINT
20 FOR I=32 TO 127
30 PRINT CHR$(I);
40 NEXT I
50 FOR J=161 TO 255
60 PRINT CHR$(J);
70 NEXT J
80 PRINT:PRINT:PRINT"**END TEST**"
```

Ne possiamo calcolare il tempo di esecuzione nel modo seguente:

1. Il tempo TI (o TIME$) viene assegnato ad una variabile A all'inizio dell'esecuzione.
2. Lo stesso TI (o TIME$) viene poi riassegnato ad un'altra variabile B alla fine dell'esecuzione.
3. Facendo la differenza tra le due variabili si ottiene il tempo impiegato per l'esecuzione del programma misurato in "attimi".

Riportiamo il programma dato più sopra con l'aggiunta delle istruzioni per calcolare il suo tempo di esecuzione:

```
Passo 1   10 PRINT"⊃**KEYBOARD TEST**":PRINT
          15 A=TI
          20 FOR I=32 TO 127
          30 PRINT CHR$(I);
          40 NEXT I
          50 FOR J=161 TO 255
          60 PRINT CHR$(J);
          70 NEXT J
Passo 2   75 B=TI
          80 PRINT:PRINT:PRINT"**END TEST**"
Passo 3   100 PRINT:PRINT"TI = ";B-A
```

Alla linea 15 il tempo TI viene assegnato alla variabile A:

```
15 A=TI
   A = TI 6001762
   A 6001762
```

Durante l'esecuzione del programma TI viene automaticamente incrementato 60 volte al secondo. Alla fine, alla linea 75, assegnamo il nuovo valore di TI a B:

```
75 B=TI
   B=TI 6001953
   B= 6001953
```

Alla linea 100 si calcola la differenza tra i due tempi A e B:

```
100 PRINT PRINT"TI=";B-A
      B 6001953
    - A 6001762
    ----------
```

Nell'esempio si vede che occorrono 191 "attimi" per visualizzare tutti i caratteri della tastiera sullo schermo. Per avere il tempo in secondi basta dividere per 60:

191/60 = 3.1833

cioè occorrono 3.1833 secondi per eseguire questo programma. Qui di seguito trovate un esempio di esecuzione:

```
**KEYBOARD TEST**

 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFG
HIJKLMNOPQRSTUVWXYZ[\]↑← !"#$%&'()*+,-./
0123456789:;<=>?▗▄▝▟ ▚▜ ▙▘▖▗▟▟▌▌ ▛▜
▟▙ ▝▘▘▜▛▌▔▔▔▌ ▙▗ ▝▟▌▛▙▟▌ ▗▟▟▙▘▘▌ ▙▜▛▌▛▜
▗▄▝▟ ▚▜ ▙▘▖▗▟▟▌▌ ▛▜▟▙ ▝▘ π

**END TEST**

TI = 191
```

Visualizzazione di un orologio digitale

Il programma che qui di seguito vi riportiamo è molto lungo, ma anche molto interessante. Esso dà l'ora in forma digitale sullo schermo usando delle grosse cifre create con i caratteri grafici. Esso dà solo le ore e i minuti perchè sullo schermo non vi è ulteriore spazio per i secondi. Notate che tale programma è quasi tutto costituito da istruzioni PRINT.

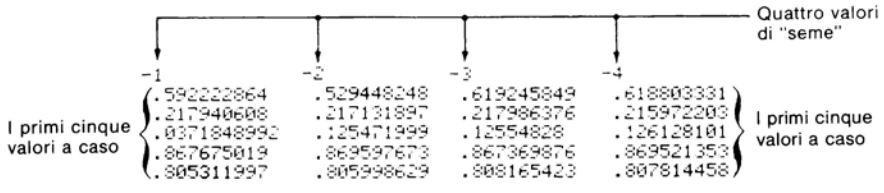
```
100 PRINT"[illegible]";
110 S=INT(TIME/60)
120 M=INT(S/60)
130 H=INT(M/60)
140 M=M-H*60
150 T=H
160 GOSUB500
170 PRINT"[illegible] [illegible] [illegible] [illegible] [illegible] ";
180 T=M
190 GOSUB500
200 PRINT"[illegible]";
210 GOTO110
500 U=T-10*INT(T/10)
510 T=INT(T/10)
520 D=T+1
530 GOSUB600
540 D=U+1
550 GOSUB600
560 RETURN
600 ON D GOSUB 1000,1100,1200,1300,1400,1500,1600,
    1700,1800,1900
610 RETURN
1000 PRINT" [illegible]";
1001 PRINT"[illegible]";
1002 PRINT"[illegible]";
1003 PRINT"[illegible]";
1004 PRINT"[illegible]";
1005 PRINT"[illegible]";
1006 PRINT"[illegible]";
1007 PRINT"[illegible]";
1008 PRINT"[illegible]";
1009 PRINT"[illegible]";
1010 RETURN
1100 PRINT"[illegible]";
1101 PRINT"[illegible]";
1102 PRINT"[illegible]";
1103 PRINT"[illegible]";
1104 PRINT"[illegible]";
1105 PRINT"[illegible]";
1106 PRINT"[illegible]";
1107 PRINT"[illegible]";
1108 PRINT"[illegible]";
1109 PRINT"[illegible]";
1110 RETURN
1200 PRINT"[illegible]";
1201 PRINT"[illegible]";
1202 PRINT"[illegible]";
1203 PRINT"[illegible]";
1204 PRINT"[illegible]";
1205 PRINT"[illegible]";
1206 PRINT"[illegible]";
1207 PRINT"[illegible]";
1208 PRINT"[illegible]";
1209 PRINT"[illegible]";
1210 RETURN
1300 PRINT"[illegible]";
1301 PRINT"[illegible]";
1302 PRINT"[illegible]";
```

```
1303 PRINT"      ◢  ▀▀▀▀▀▀▀▀▀▀";
1304 PRINT"    ▞    ▟▀▀▀▀▀▀▀▀▀▀";
1305 PRINT"    ▞    ▜▀▀▀▀▀▀▀▀▀▀";
1306 PRINT"      ◥▞  ▀▀▀▀▀▀▀▀▀▀";
1307 PRINT"▞  ▜  ◢  ▀▀▀▀▀▀▀▀▀▀";
1308 PRINT"◥▞      ▟▀▀▀▀▀▀▀▀▀▀";
1309 PRINT" ◥▞    ▟   ▄▄▄▄▄▄▄▄▄▄";
1310 RETURN
1400 PRINT"    ◢ ▟  ▌▌▌▌▌▌▌▌▌▌";
1401 PRINT"   ◢  ▟  ▌▌▌▌▌▌▌▌▌▌";
1402 PRINT"  ◢   ▟  ▌▌▌▌▌▌▌▌▌▌";
1403 PRINT" ◢    ▟  ▌▌▌▌▌▌▌▌▌▌";
1404 PRINT"◢  ▟◥▞  ▟  ▌▌▌▌▌▌▌▌▌▌",
1405 PRINT"▞  ▟ ▞  ▟  ▌▌▌▌▌▌▌▌▌▌";
1406 PRINT"▞        ▀▀▀▀▀▀▀▀▀▀";
1407 PRINT"▞        ▀▀▀▀▀▀▀▀▀▀";
1408 PRINT"    ▞  ▟  ▌▌▌▌▌▌▌▌▌▌";
1409 PRINT"    ▞  ▟   ▄▄▄▄▄▄▄▄▄▄";
1410 RETURN
1500 PRINT"▞        ▀▀▀▀▀▀▀▀▀▀";
1501 PRINT"▞        ▀▀▀▀▀▀▀▀▀▀";
1502 PRINT"▞  ▟      ▌▌▌▌▌▌▌▌▌▌";
1503 PRINT"▞      ▜ ▌▌▌▌▌▌▌▌▌▌";
1504 PRINT"▞      ▜▀▀▀▀▀▀▀▀▀▀";
1505 PRINT"      ◥▞  ▀▀▀▀▀▀▀▀▀▀";
1506 PRINT"      ▞  ▀▀▀▀▀▀▀▀▀▀";
1507 PRINT"▞  ▜  ◢  ▀▀▀▀▀▀▀▀▀▀";
1508 PRINT"◥▞      ▟▀▀▀▀▀▀▀▀▀▀";
1509 PRINT" ◥▞    ▟   ▄▄▄▄▄▄▄▄▄▄";
1510 RETURN
1600 PRINT" ◢    ▜ ▌▌▌▌▌▌▌▌▌▌";
1601 PRINT"◢      ▜▀▀▀▀▀▀▀▀▀▀";
1602 PRINT"▞  ▟  ◥▞  ▀▀▀▀▀▀▀▀▀▀";
1603 PRINT"▞  ▟      ▌▌▌▌▌▌▌▌▌▌";
1604 PRINT"▞      ▜ ▌▌▌▌▌▌▌▌▌▌";
1605 PRINT"▞      ▜▀▀▀▀▀▀▀▀▀▀";
1606 PRINT"▞  ▟  ◥▞  ▀▀▀▀▀▀▀▀▀▀";
1607 PRINT"▞  ▜  ◢  ▀▀▀▀▀▀▀▀▀▀";
1608 PRINT"◥▞      ▟▀▀▀▀▀▀▀▀▀▀";
1609 PRINT" ◥▞    ▟   ▄▄▄▄▄▄▄▄▄▄";
1610 RETURN
1700 PRINT"▞        ▀▀▀▀▀▀▀▀▀▀";
1701 PRINT"▞        ▀▀▀▀▀▀▀▀▀▀";
1702 PRINT"      ◢  ▀▀▀▀▀▀▀▀▀▀";
1703 PRINT"    ◢  ▟▀▀▀▀▀▀▀▀▀▀";
1704 PRINT"   ◢  ▟ ▌▌▌▌▌▌▌▌▌▌";
1705 PRINT"  ◢  ▟  ▌▌▌▌▌▌▌▌▌▌";
1706 PRINT"  ▞  ▟  ▌▌▌▌▌▌▌▌▌▌";
1707 PRINT"  ▞  ▟  ▌▌▌▌▌▌▌▌▌▌";
1708 PRINT"  ▞  ▟  ▌▌▌▌▌▌▌▌▌▌";
1709 PRINT"  ▞  ▟     ▄▄▄▄▄▄▄▄▄▄";
1710 RETURN
1800 PRINT" ◢    ▜ ▌▌▌▌▌▌▌▌▌▌";
1801 PRINT"◢      ▜▀▀▀▀▀▀▀▀▀▀";
1802 PRINT"▞  ▟  ◥▞  ▀▀▀▀▀▀▀▀▀▀";
1803 PRINT"▞  ▜  ◢  ▀▀▀▀▀▀▀▀▀▀";
1804 PRINT"◥▞      ▟▀▀▀▀▀▀▀▀▀▀";
1805 PRINT"◢      ▜▀▀▀▀▀▀▀▀▀▀";
1806 PRINT"▞  ▟  ◥▞  ▀▀▀▀▀▀▀▀▀▀";
```

```
1807 PRINT"▚  ▜  ▚  ▟▀▀▀
```

lunga sequenza e non solo pochi esempi. Questa sequenza varia da calcolatore a calcolatore, ma è sempre la stessa per una specifica macchina.

Ogni sequenza di numeri a caso è identificata da un seme che è sempre un numero negativo. Ecco uno schema esemplificativo:



Ad ogni numero seme corrisponde una diversa sequenza. Siccome i numeri negativi sono innumerabili anche le sequenze possibili sono innumerabili.

Per specificare una sequenza dovete porre il numero seme negativo come argomento della funzione RND. Questa funzione può essere poi assegnata ad una qualunque variabile:

```
20 X=RND(-2)
```

Attenzione però che porre l'argomento negativo nella funzione RND equivale a ri-inizializzarla, cioè se la stavate già usando così facendo ritornate da capo. Se per esempio in un particolare calcolatore CBM il primo numero della sequenza a caso con seme −2 è .529448248 allora ogni volta che ponete RND (−2) ottenete questo stesso valore iniziale. Ricordate però che con un qualunque altro calcolatore CBM non otterrete lo stesso numero iniziale pur dando il seme −2.

Sequenza di numeri a caso

Una volta che avete inizializzato la funzione RND potete ottenere i valori della sua sequenza semplicemente richiamandola con un qualunque argomento positivo. In questo esempio richiamiamo cinque diverse sequenze caratterizzate dai semi −1, −2, −3, −4 e −5. Per ogni sequenza visualizziamo i primi cinque valori:

```
30 FOR I=-1 TO -5 STEP -1
35 X=RND(I):PRINTI
40 FOR J=1 TO 5
50 PRINT RND(1)
60 NEXT J
70 NEXT I
100 STOP
```

La funzione RND posta alla linea 35 inizializza per cinque volte la generazione dei

numeri a caso. La RND della linea 50 genera invece i singoli valori delle cinque sequenze. Ecco il risultato:

```
-1
 .592222864
 .217940608
 .0371848992
 .867675019
 .805311997
-2
 .529448248
 .217131897
 .125471999
 .869597673
 .805998629
-3
 .619245849
 .217986376
 .12554828
 .867369876
 .808165423
-4
 .618803331
 .215972203
 .126128101
 .869521353
 .807814458
-5
 .529738186
 .216918235
 .128416908
 .868422708
 .717787951
```

Per dimostrarvi che la sequenza di numeri è sempre la stessa per ogni fissato seme, vi proponiamo questo programma in cui le sequenze sono interrotte al terzo valore e poi riprese fino al quinto:

```
30 FOR I=-1 TO -5 STEP -1
35 X=RND(I): PRINT I
40 FOR J=1 TO 3
50 PRINT RND(1)
60 NEXT J
70 FOR K=10 TO 11
75 PRINT RND(1)
80 NEXT K
90 NEXT I
100 STOP
```

Ponete in esecuzione questo programma e vedrete che le sequenze sono esattamente le stesse del caso precedente. Questo significa che una funzione RANDOM continua a dare il valore a caso successivo ogni qualvolta, e in qualunque modo, sia richiamata. Solo quando il suo argomento è negativo allora ritorna al primo valore della sequenza. Per esempio se avessimo aggiunto questa istruzione al nostro programma:

```
65 X=RND(I)
```

avremmo ottenuto che, dopo aver visualizzato i primi tre valori, i successivi due sarebbero stati eguali al primo e al secondo.

Provate ora per esercizio a variare i parametri di quest'ultimo programma oppure generate la sequenza di numeri a caso che ritenete più interessante.

Stampa di numeri a caso

Per meglio comprendere e confrontare tra loro le sequenze dei numeri a caso è preferibile che voi li stampiate piuttosto che visualizzarli. (Ovviamente ciò implica che abbiate una stampante). Nel capitolo 6 vedremo come programmare una stampante, ma ora è sufficiente che vediate in maniera intuitiva le istruzionei necessarie per il suo funzionamento:

```
10 OPEN 4,4
20 CMD 4
30 FOR I=-1 TO -5 STEP -1
35 X=RND(I):PRINTI
40 FOR J=1 TO 5
50 PRINT RND(1)
60 NEXT J
70 NEXT I
80 PRINT#4
90 CLOSE 4
100 STOP
```

Le istruzioni alle linee 10 e 20 aprono il collegamento con la stampante mentre quelle alle linee 80 e 90 lo chiudono. Eseguite quindi questo programma così avrete su carta le sequenze di numeri a caso e potrete più facilmente confrontarle tra loro.

Se l'argomento della funzione RND non è ne positivo ne negativo, cioè è posto eguale a zero, allora si ottiene una sequenza di valori che dipende dall'orologio del sistema. Tali sequenze non saranno mai eguali, ma sempre molto simili. Provate infatti a sostiuire l'argomento della RND, alla linea 50, con 0 invece di 1 ed eseguite più volte il programma. Otterrete sequenze simili, ma mai eguali.

Uso di "semi" a caso

Per generare sequenze, sempre completamente diverse, dovete in qualche modo usare dei numeri seme presi a caso. Questo può essere fatto usando il valore TI dell'orologio del calcolatore:

```
10 X=RND(-TI): REM START SEED
```

Adesso avrete una diversa sequenza ogni volta che eseguirete l'istruzione 10 (escluso il caso che la eseguiate esattamente ogni 24 ore!).

Un altro modo per generare un seme a caso è quello di scrivere RND (−RND(0)), ma solo se il vostro calcolatore ha le nuove ROM per il BASIC. Per esempio:

```
10 X=RND(-RND(0))
```

Anche qui avrete una diversa sequenza ogni volta che questa istruzione viene eseguita.

Nel programma successivo abbiamo usato l'argomento −TI perchè è compatibile sia con le vecchie che con le nuove ROM, ma se potete è preferibile usare l'argomento −RND(0) invece di −TI.

Sequenza analoga al lancio dei dati

I valori ottenuti con la funzione RND sono sempre compresi tra 0 e 1. Se volete numeri compresi in un altro intervallo, dovete allora fare una conversione. Se per esempio volete ottenere solo i sei numeri interi da 1 a 6, come nel caso del lancio di un dado, dovete fare la seguente conversione. Moltiplicate dapprima RND per 6:

```
6•RND(1)
```

questo vi darà i numeri frazionari tra 0 e 6 ($0 < n < 6$). Aggiungete quindi 1 per avere $1 < n < 7$:

```
6•RND(1)+1
```

Convertite quindi il numero da frazionario in intero togliendogli la parte decimale; otterrete infine i numeri interi tra 1 e 6:

```
  INT(6•RND(1)+1)
o :
  A%=6•RND(1)+1
```

In generale se volte generare a caso numeri interi compresi in un certo intervallo potete usare queste istruzioni:

INT((n+1)•RND(1))	Intervallo tra 0 e n
INT(n•RND(1)+1)	Intervallo tra 1 e n
INT((n−m+1)•RND(1)+m	Intervallo tra m e n

Notate però che i numeri interi nel nostro calcolatore possono essere compresi solo nell'intervallo ± 32767.

Il programma seguente genera a caso numeri interi comrpesi tra −50 e 50 (vedi linea 10). Il seme di partenza è preso come valore −TI per cui ogni volta che il programma è rieseguito genera sequenze diverse. I numeri possono essere anche negativi perchè sono compresi tra + e − 50. Per formare il programma dovete premere STOP. Ecco il programma e un modulo di esecuzione:

```
10 M=-50 N=50
20 X=RND(-TI):PRINT X
30 FOR I=1 TO 8
40 C%=(N-M+1)*RND(1)+M
50 PRINT C%; NEXT I
60 PRINT:GOTO 30
RUN
```

```
 3.27633085E-06
-14  9 -34 -35 -47 -44  28  31
 29 -8 -36 -28 -42 -28  15  14
 7 -13  3 -8  8  41  19 -43
 35  12  24 -7 -7 -21 -47  1
-32 -49  7 -49  28 -22 -17 -24
-12  7  27  1  11  9 -18  35
 48  49  1  34 -46 -29 -43  29
-18  5 -30  2  8 -28 -13 -23
 48 -15 -12 -45  26  44 -25  2
-9  4  27  50  33 -16 -43 -15
 20  20  17  43 -18 -48 -38  24
-16  43 -50  36 -38  5  11  25
-30  6 -25 -47  32  10  42 -21
-47 -38 -28 -8  16 -20  42 -4
-34  36 -17  27 -8 -49 -6 -35
-19  19 -35  48 -42  36 -25  2
-49  37  47  38 -20 -25  32 -50
-5 -35 -35  17 -41  36 -19  4
 33 -20  45 -7  48 -4 -33 -10
 1  27 -39 -14 -38 -6  4  10
-5  17  2  49  0 -40 -5  32
-50  32 -24 -37 -38  22 -13 -27
-24 -30  35  10  6  16 -50  49
-49  50  43  38 -21  47 -43  28
 32 -35 -18 -5  27 -46 -14  23
-49 -45  27  7 -35  1  46 -25
-8  20 -8 -12 -46 -31 -17 -18
-47  47 -49  18  47  17  40 -13
-40  48 -41 -33  5 -14 -46  45
-29 -37  22  17  42  33 -31  49
 8 -4  36  37  11  18  29  25
 0 -1  2 -16  32 -29 -31  33
-9 -41 -4  47  12 -22  9 -48
-40  32  15  32 -50  3 -9  19
```

Provate a cambiare i valori M e N alla linea 10 per esercizio. In particolare ponete M = 1 e N = 6 così che ritrovate la sequenza analoga al lancio di un dato.

Selezione a caso delle carte da gioco

Se osservate attentamente la sequenza precedente di numeri compresi nell'intervallo ± 50 vi accorgerete che al massimo dopo 100 valori i numeri devono sicuramente ripetersi. Cioè uno stesso valore può uscire ripetute volte e anche di seguito. **Per molte applicazioni queste sequenze di numeri generati a caso con ripetizione vanno bene, ma ci sono altri casi in cui i numeri non si devono ripetere.** Un esempio classico è quello delle carte da gioco in cui se una carta è stata presa del mazzo non può più essere ripescata.

Vediamo come è possibile scrivere un programma per generare numeri a caso senza ripetizioni. Per fare un esempio concreto supponiamo di simulare la successione di 52 carte da gioco prese da un mazzo ben mescolato. Identifichiamo il valore delle carte in un vettore D% di 52 elementi (D%(0) non viene usato). Questa attribuzione può essere fatta come preferite e supponiamo sia la seguente:

Asso = 1, 2 = 2, 3 = 3, ... Q = 12, K = 13

Spade = da 0 Cuori = da 13 Fiori = da 26 Quadri = da 39

Con questa convenzione abbiamo l'asso di picche = 1 + 0 = 1, la regina di picche = 12 + 0 = 12, il tre di cuori = 3 + 13 = 16, ecc.

Per tenere conto delle carte già uscite usiamo un vettore flag FL i cui 52 elementi possono essere 0 o 1. Il valore di seme viene visualizzato alla linea 20. Successivamente vengono visualizzati i 52 valori della sequenza non ripetitiva, come potete voi stessi controllare. Ogni volta che rieseguite il programma ottenete una sequenza diversa.

```
10 DIM FL(52),D%(52)
20 X=RND(-TI):PRINT X
30 FOR I=1 TO 52
40 C%=52*RND(1)+1
50 IF FL(C%)<>0 GOTO 40
60 D%(I)=C%:FL(C%)=1
70 PRINT C%;
80 NEXT I
RUN
 1.18586613E-05
 48  40  13  37  50  43  46  31  49  44
 23  38  25  11  9  35  32  30  24  41
26  5  6  1  45  10  21  14  42  20  15
 34  18  52  47  7  16  8  19  33  36  4
  17  3  22  27  29  28  39  2  51  12
RUN
 1.01154728E-06
 14  35  52  50  26  48  27  36  34  25
 18  20  41  33  39  7  46  24  23  28  1
9  3  12  43  2  31  44  4  1  32  37  3
0  40  22  45  48  42  49  16  11  6  10
  29  9  51  17  8  15  38  5  21  13
```

Se avete effettivamente eseguito questo programma avrete notato che diviene più lento man mano che "estrae le carte". Questo succede perchè deve estrarre più numeri a caso prima di trovarne uno non ancora estratto. Se volete potete voi stessi migliorare questa routine in modo da renderla più veloce: per esempio potete far uscire l'ultima carta, proprio perchè è l'ultima rimasta, senza attendere che sia generata dai numeri a caso.

CARATTERI FORZATI A CASO SULLO SCHERMO CON POKE

Il programma che segue è una modifica del programma PROVA. Invece di visualizzare un carattere in maniera uniforme e continua su tutto lo schermo, esso lo fa apparire a caso in un punto qualunque mediante il comando POKE indirizzato a una delle 1.000 posizioni della memoria dello schermo.

Il programma è questo:

```
10 REM   **********  P R O V A  **********
15 REM
20 REM    VISUALIZZAZIONE A CASO DI UN
30 REM    CARATTERE BATTUTO ALLA TASTIERA
35 REM
40 REM   ********************************
90 PRINT"BATTI UN TASTO O <R> PER TERMINARE";
100 GETC$: IF C$="" GOTO 100
105 IF C$=CHR$(13) GOTO 170
110 PRINT"□";          'REM PULISCE LO SCHERMO
120 X=RND(-TI)         'REM PONE UN NUOVO "SEME"
125 C=(ASC(C$)AND128)/2 OR (ASC(C$)AND63)
127 A=1000*RND(1)+32768
130 POKE A,C           'REM VISUALIZ. CARATTERE
140 GET D$: IF D$="" GOTO 127
150 C$=D$
160 GOTO 105
170 END
```

Fino alla linea 110 esso è uguale alla versione precedente; carica un nuovo valore di C$ e pulisce lo schermo. Alla linea 120 viene preparato un seme per la successiva generazione a caso dei punti dello schermo. Alla linea 125 C$ è convertito nell'equivalente numero POKE. L'istruzione 127 calcola un indirizzo dello schermo a caso, nell'intervallo tra 32768 e 33767, mediante la routine già vista prima con M = 32768 e N = 33767:

(n−m+1)•RND(1)+m	Formula per
(33767-32768+1)•RND(1)+32768	la linea 127
=1000•RND(1)+32768	

Notate che non possiamo usare variabili numeriche intere, ne con la funzione INT ne con il segno %, per indicare l'indirizzo sullo schermo perchè essi sono superiori al numero intero massimo 32767. Fortunatamente la funzione POKE accetta numeri frazionari che poi converte in interi troncando la parte frazionaria. (In altri casi in cui avete bisogno di numeri interi a caso fuori dall'intervallo consentito potete usare variabili frazionarie purchè esse abbiano la stessa parte intera).

A questo punto dobbiamo farvi notare che la sequenza di numeri a caso usata è del tipo con ripetizioni per cui può essere indirizzato uno stesso punto dello schermo più volte. Inoltre occorrerà attendere sempre più tempo per riempire le ultime posizioni che rimangono libere. La prima metà dello schermo viene riempita quasi subito, ma per riempirlo tutto passeranno anche più di tre minuti.

Un modo per accelerare il nostro programma consiste nell'eliminare i comandi POKE indirizzati alle posizioni dello schermo già coperte.

In questa seconda versione di PROVA si fa in modo di generare, ad ogni passo, numeri a caso in un intervallo più piccolo così da lasciare fuori i valori già usciti. Per poter far questo si deve introdurre una tavola di corrispondenza T() di 1000 elementi e ad ogni passo si spostano verso la sua parte alta i valori già usciti, mentre

si usa la parte bassa come nuovo intervallo per generare il numero successivo. Il programma PROVA versione 2 è il seguente:

```
5 REM    VERSIONE 2 RANDOM
6 REM
10 REM   **********  P R O V A  **********
15 REM
20 REM    VISUALIZZAZIONE A CASO DI UN
30 REM    CARATTERE BATTUTO ALLA TASTIERA
35 REM
40 REM   ********************************
70 DIM T(999)
80 GOSUB 200          :REM INIZIZLIZZ. TAVOLA
90 PRINT"BATTI UN TASTO O <R> PER TERMINARE";
100 GETC$: IF C$="" GOTO 100
105 IF C$=CHR$(13) GOTO 170
110 PRINT"□";         :REM PULISCE LO SCHERMO
120 X=RND(-TI)        :REM PONE UN NUOVO "SEME"
125 C=(ASC(C$)AND128)/2 OR (ASC(C$)AND63)
126 FOR N=999 TO 0 STEP -1
127 A%=(N+1)*RND(1)   :REM PRENDE UN ELEM.
128 A=T(A%)+32768     :REM INDRIZZO POKE
129 TP=T(A%): T(A%)=T(N): T(N)=TP
130 POKE A,C
140 NEXT N
150 C$=D$
160 GOTO 100
170 END
199 REM    *** SUBROUTINE INIZIAL. TAVOLA ***
200 FOR I=0 TO 999 :T(I)=I : NEXT
210 RETURN
```

Alla linea 70 si dimensiona la tavola T() con 1000 posizioni.

Alla linea 80 si richiama una subroutine di inizializzazione di T così da ottenere: T(0) = 0, T(1) = 1, ..., T(999) = 999. È da notare che questo riempimento ordinato di T non è strettamente necessario in quanto i suoi elementi verranno poi scelti a caso.

Le linee da 90 a 125 sono le stesse della versione precedente. Con il ciclo da 126 a 140 vengono generati numeri in intervalli sempre più piccoli (STEP −1), mentre i valori già usciti sono trasferiti nella parte alta della tavola con l'istruzione 129.

Questo manuale, in due volumi, è la versione italiana del famosissimo testo americano "PET/CBM Personal Computer Guide". La scelta di presentare l'opera in due volumi è dovuta all'ampiezza e profondità degli argomenti trattati.
Tutto ciò che è necessario sapere sui calcolatori CBM si trova in questo manuale, come programmarli in BASIC, come installarli e molte altre utilissime notizie che riguardano la più famosa famiglia di Personal Computer.
Anche se questo manuale non tratta espressamente dei più recenti VIC 20 e C 64, ne consigliamo egualmente la consultazione ai possessori di questi calcolatori in quanto il linguaggio BASIC CBM è praticamente eguale per tutti i calcolatori Commodore.

L. 20.000 Cod. 332 P ISBN 88-7056-029-5

GRUPPO
EDITORIALE
JACKSON

Adam Osborne
Carrol S. Donahue

PET/CBM GUIDA ALL'USO VOL.1

96